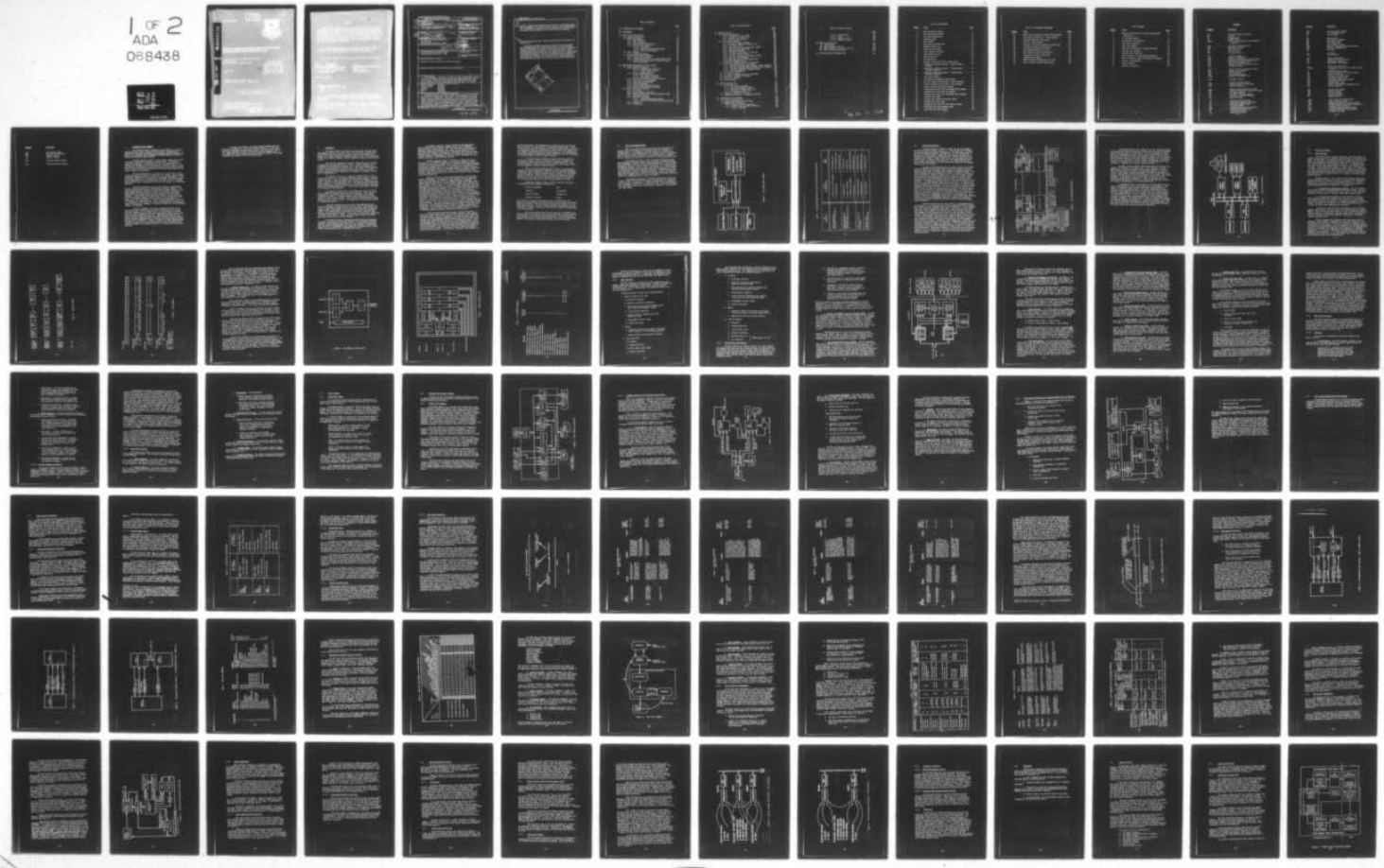
AD-A068 438 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/3
DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEM--ETC(U)
MAR 79 R C MASON, T R PRICE, B A RICH F33615-75-C-1178
UNCLASSIFIED AFAL-TR-79-1027 NL

1 OF 2
ADA
088438



LEAB

[redacted]

[redacted]

TRW DEFENSE AND SPACE SYSTEMS GROUP, INC.
ONE SPACE PARK
REDONDO BEACH, CALIFORNIA 90278

[redacted]

TECHNICAL REPORT AFAL-TR-79-1077
Final Report for the Period April 1975 – September 1978

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAL-TR-79-1027	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEMONSTRATION,	5. TYPE OF REPORT & PERIOD COVERED: FINAL REPORT, 28 Apr 75 - 30 Sept 78
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): R. C. Mason, R. J. Slightam, T. R. Price, J. A. Stautberg, B. A. Rich, W. P. Whalen, C. E. Wilent	8. CONTRACT OR GRANT NUMBER(s): F33615-75-C-1178
9. PERFORMING ORGANIZATION NAME AND ADDRESS: TRW Defense and Space Systems Group, One Space Park, Redondo Beach, California 90278	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 2052-05-01
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: Mar 79
	13. NUMBER OF PAGES: 156
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Air Force Avionics Laboratory (AFAL), Wright-Patterson AFB, Ohio 45433	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The development, integration, and test of DAIS and the support facility leading to the highly successful real-time DAIS Demonstration was a result of an excellent cooperative effort between AFAL DAIS personnel and TRW System Integration and Test Coordination (SITC) personnel. This report

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Avionics Software	Displays
Avionic Systems	Hot Bench
Computers	Multiplex
Controls	Processor
DAIS	Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Digital Avionics Information System (DAIS) has been characterized as a system architecture which can be applied and configured for a broad class of avionic applications and missions. The DAIS concept, therefore, proposed that the processing, information transfer, and the control and display functions or core elements be common and service the avionic application functional areas on an integrated basis. These fundamental system characteristics, along with the DAIS system attributes, such as modifiability and modularity, are presented in this report. The specific system → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409 637

18.

provides a complete and comprehensive description of DAIS architecture and core elements, the hot bench support facility, and the demonstration. Thus, recognition and appreciation are extended to AFAL DAIS personnel for specific inputs included in this report to make it complete.

20.

features which provide these system characteristics and attributes are described.

The DAIS core elements along with support hardware and software were integrated into a hot bench support facility. A representative avionics application and real-time mission were performed to demonstrate the DAIS architecture and concepts. This report describes the DAIS architecture and core elements, and the support facilities elements that were integrated together to perform the successful mission demonstration. The report also describes how the basic DAIS attributes and features facilitated the integration and testing as well as system adaptability and flexibility to a broad class of avionic applications and missions.

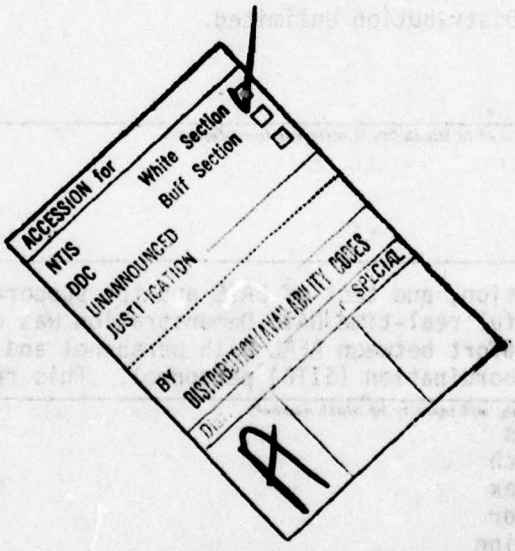


TABLE OF CONTENTS

TABLE OF CONTENTS(CON'T)

TABLE OF CONTENTS CONCLUDED

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONCLUDED)

LIST OF TABLES

GLOSSARY

ACRONYM	DEFINITION
ADI	Attitude Direction Indicator
Alt.	Altitude
AP	Armament Panel
AR	Aiming Reticle
AVSAIL	Avionic System Analysis and Integration Laboratory
BCIU	Bus Control Interface Unit
BCM	Bus Control Module
BFL	Bomb Fall Line
BIT	Built-In-Test
BMU	Bus Monitor Unit
CAS	Close Air Support
C&D	Controls and Displays
CBIS	Controls and Backup Instruments System
CCIP	Continuously Computed Impact Point
CIU	Console Intelligence Unit
CPCI	Computer Program Configuration Item
CRT	Cathode Ray Tube
DAIS	Digital Avionics Information System
DEK	Digital Entry Keyboard
DISP	Display Process
DMA	Direct Memory Access
DS/MU	Display Switch/Memory Unit
EQUIP	Equipment Process
FIM	Facility Interface Module
FLIR	Forward Looking Infrared Radar
FPM	Flight Path Marker
HARS	Heading Attitude Reference System
HAS	Hardware Architecture Simulation System
HBC	Hot Bench Computer
HSD	Horizontal Situation Display
HSI	Horizontal Situation Indicator
HUD	Head Up Display
ICD	Interface Control Document
ICS	Interpretive Computer Simulation
ILS	Instrument Landing System
IM	Interface Module
IMFK	Integrated Multifunction Keyboard
INS	Inertial Navigation System
IP	Initialization Point
ITB	Integrated Test Bed

ACRONYM	DEFINITION
LDGP	Low Drag General Purpose
LRU	Line Replaceable Unit
LOS	Line of Sight
M/E	Message Error
MFK	Multifunction Keyboard
MMP	Master Mode Panel
MMU	Mass Memory Unit
MPD	Multi-Purpose Display
MPDG	Modular Programmable Display Generator
MTU	Multiplex Terminal Unit
MUX	Multiplex
NAV	Navigation
N.M.	Nautical Miles
OAP	Offset Aiming Point
OFP	Operational Flight Program
OPS	Operational Sequencer
OTP	Operational Test Program
PAL	PALEFAC Auxiliary File
PALEFAC	Partitioning Analyzing and Linkage Editing Facility
PCP	Processor Control Panel
PDS	Power Distribution and Control System
PGI	PALEFAC Global Unit
PIM	Processor Interface Module
PIO	Programmed Input/Output
PMC	Performance Monitor and Control
PMD	PALEFAC Mission File
PMI	PALEFAC Module Input
PPI	PALEFAC Partitioning Information Files
PRE	Post Run Editor
RAM	Random Access Memory
RAT	Ram-Air Turbine
ROM	Read Only Memory
ROT	Rough Output Tape
RT	Remote Terminal
SCADU	Super Control and Display Unit
SCU	Sensor Controller Unit
SDVS	Software Design & Verification System
SITC	System Integration and Test Coordination
SLS	Statement Level Simulation
SSDF	Simulated Subsystem Data Formatter
SSIU	Simulated Subsystem Interface Unit
STS	Software Test Stand

ACRONYM	DEFINITION
TCC	Test Control Center
TDM	Time Division Multiplex
T/F	Terminal Failure
T/R	Transmit Receive
URT	Universal Remote Terminal
VSD	Vertical Situation Display

1. INTRODUCTION AND SUMMARY

The Digital Avionics Information System (DAIS) is a system architecture for avionic systems utilizing digital technology to reduce life cycle costs by defining and developing modular hardware and software core elements and standardized interfaces which can be configured and applied to many aircraft.

The DAIS approach reflects a total system concept rather than a functional subsystem or hardware oriented system. For example, a "Navigation Subsystem" in DAIS does not refer to a dedicated set of hardware and software which perform only the navigation function. DAIS architecture and elements are not dedicated to any one avionic function, but are used to perform and integrate the functions associated with the avionic sensors and subsystems.

The basic architecture is designed for a broad class of configurations where the number of processors, for example, can be reduced or enlarged depending upon the avionics and mission requirements. Standardization, modularity, and application independent executive software allows adaptability of this architecture to a broad class of different applications as well as making mission-to-mission changes in a particular aircraft.

The DAIS architecture consists of processors communicating with each other and the other system elements (sensors, weapons, and controls and displays) through a standardized multiplex data bus. This standardized multiplex data bus provides dual redundant information paths between the system resources (each computer and other system elements). Centralized system single-point control is performed by a processor resident software executive that can be relocated for redundancy. Application software is structured to provide modularity, reliability, and transferability. This system architecture is flexible to accommodate a wide variety of avionics configurations, missions, and sensors; and provides redundancy to improve availability, and accommodate changes in technology.

The basic elements of the DAIS architecture which can be restructured for various aircraft avionic configurations are called DAIS core elements (or building blocks) and are composed of the DAIS multiplex, DAIS processors (with associated memory), DAIS mission software, and DAIS controls and displays. Additional elements are the support software elements, namely the JOVIAL Compiler, Software Design and Verification System (SDVS) and Partitioning, Analyzing and Linkage Editing Facility (PALEFAC). Sensors, weapons, and other subsystems are selected as required for the particular mission and connected to the interface modules of the Remote Terminals or directly to the multiplex data bus.

The DAIS core elements along with support hardware and software were integrated into a hot bench support facility. A real-time mission was performed to demonstrate the DAIS architecture and concepts. This report describes the DAIS architecture and core elements, and the support facilities elements that were integrated together and tested to perform the successful mission demonstration.

2. BACKGROUND

The Digital Avionics Information System (DAIS) has been under development by the Air Force since 1973. DAIS provides a system architecture which can adapt currently available avionics subsystems and sensors to a common, modularly expandable avionics core. The architecture of DAIS also provides a general system framework to design future avionics systems.

The classical approach to avionics system design has provided a system unique to each particular aircraft. Avionics systems tend to be snapshots of technology available at the time of development with little or no relationship to previous work. The result of this approach is that aircraft which are part of the operational fleet at a given point in time have neither commonality of hardware or software, nor any sort of compatibility.

The effect of the lack of commonality/compatibility is felt in cost and reliability of the newly developed weapon system or upgraded/retrofit efforts. Since new hardware and software developments are required, additional front end design costs are incurred. The most serious effect, however, is the maintenance cost. Different hardware across the fleet compounds the problems of spares provisioning and repair, personnel training, and maintenance support. These factors contribute to the high cost of ownership for avionics.

In addition, since avionic procurements are relatively small lots, hardware production maturity is rarely reached. This results in hardware with lower MTBF's than commercial counterparts with larger production quantities. Although software accomplishes much the same function on all military aircraft, the classical approach results in new design for each aircraft system. This generates a corresponding problem of software reliability.

The above problems are not the only ones generated by the classical approach. Perhaps the most serious, from an operational viewpoint, is the limited adaptability of point designed avionic systems to changing threat environments or expanded mission roles for the weapon system. Past experience has shown that the airframe lifetime is significantly longer than the useful (in the sense of operational capability) lifetime of the avionics. Hence, a typical weapon system is modified many times during the life of the airframe.

The Digital Avionics Information System (DAIS) was conceived as a means of avoiding the problems inherent in the classical avionics approach. DAIS considers the avionics job as a hybrid of process control and information management functions, and provides a system architecture which is independent of the particular airframe and, to a large extent, the technology used to implement the system.

For example, tactical aircraft accomplish the same generic mission functions of navigation, communication, stores management, weapon delivery, flight control, etc. The classical avionics approach has been to physically partition the system in accordance with the functional partitioning of the mission. Therefore, navigation systems, weapon delivery systems, and flight control systems have resulted. Each of these systems is independent of the other and is developed by independent contractors using different designs.

The DAIS approach views the avionics as a whole and not as a collection of prior defined systems. By using this approach, common functions of the avionics system could be identified which were required by each of the mission functions. This resulted in an integrated avionics system which accomplishes all of the required mission functions but is partitioned along the lines of processing hardware, software, information transfer, and display and control. Hence, a new partitioning of the avionics system was defined so that the mission functions could be mapped onto that partitioning.

The next step was to apply some constraints to the evolving system architecture. For reasons of cost (acquisition and especially life cycle), maximum utilization of common hardware was desired. If common computers are used in a multi-computer system, development costs can be saved, software costs can be reduced (common code), and maintenance problems can be reduced. In order to enhance redundancy, and error management and recovery, maximum sharing of information and maximum use of shared resources were desired. This was based on the premise that if the necessary information is available system wide, the system can reorient the task of selected resources to accomplish critical mission functions when primary resources have failed. One of the few performance improvement constraints was to provide a better method of interfacing to the pilot. The classical system forces the pilot to process large amounts of raw data as well as make decisions. The DAIS approach was to process data for the pilot and aid his decision process, thereby reducing pilot workload. The final constraint was to provide an open ended system capability. The architecture chosen should be capable of handling tasks larger than any anticipated today and should not depend upon today's technology. Such a system should be able to expand or change by adding or modifying resources without affecting the architecture.

With the above constraints defined, the next step was to define the system architecture. Using terms analogous to the computer literature usage, system architecture is the way the system appears to the user while the topology (organization) is the way the various subsystems are interconnected logically and physically to form the system. The topology chosen for DAIS was that of a distributed system. This allowed physically distributing the resources to enhance survivability and provided a means for expanding system capability without regard to physical placement of the resources required to expand the capability. The architecture provides a hierarchial system structure operating under centralized control. To the "user", therefore, DAIS appears as a

centralized system. The implementation of DAIS required the development of certain "building blocks" to achieve the design goals. These building blocks are called the DAIS core elements and consist generically of processors, multiplex information transfer, control and display and the software associated with the flight processors. The interfaces to the external world are through the sensors/subsystems and are interfaced to the core elements to make a complete system.

The DAIS concept, therefore, proposes that the processing, information transfer, control and display functions be common and service the avionics functional areas on an integrated basis. Thus, the DAIS approach reflects a total system concept rather than a functional subsystem or hardware oriented system.

AFAL initiated the DAIS program in 1973 with two separate contracts to Texas Instruments (F33615-73-C-1156) and General Dynamics (F33615-73-C-1244) to define and provide the guidelines for the design of the DAIS system. Following these studies, contracts to The Boeing Company (F33615-74-C-1108) and Texas Instruments (F33615-74-C-1023) were let to provide the initial hardware and software specifications for the DAIS core elements, and initial system designs for DAIS.

Subsequent to these studies, AFAL let contracts to design and develop the core elements as shown below:

Multiplex Equipment	IBM
Processor	Westinghouse
Mission Software	Intermetrics
Controls & Displays	Hughes

Also, AFAL let a System Integration and Test Coordination (SITC) contract with TRW Defense and Space Systems to support AFAL in combining these core elements, along with support hardware and software into an integrated hot bench. The hot bench would have the capability to simulate close air support missions in real time to evaluate and demonstrate the DAIS technology.

The initial part of the SITC effort was the development and demonstration of the Hardware Architecture Simulation, a DAIS prototype, to investigate and verify the DAIS architecture, and gain technical experience which was directly transferable to the full DAIS development effort. This effort is described in Section 8.0 of this report.

3.0 DAIS SYSTEM ARCHITECTURE

DAIS is a system architecture which can be configured for various avionic applications and missions using core elements or building blocks. The purpose of the DAIS concept is to reduce the proliferation and non-standardization of aircraft avionics, and permit the Air Force to assume initiative in the specification of standard avionic systems and interfaces for future Air Force system acquisitions.

Historically, avionic systems have been established along semi-autonomous functional areas such as navigation, weapon delivery, stores management, flight control, communications, etc. Each of these functional areas would have a digital system with its own processing, information transfer, control inputs, and display set. There has been an interface between each functional area only as necessary for interaction purposes, using non-standard interfaces. The DAIS concept proposes that the processing, information transfer, control and display functions be common and service all the previously described functional areas on an integrated basis as shown in Figure 1.

The general objectives of the DAIS architectural design are listed in Table 1 along with the design considerations or attributes to help satisfy the objectives. These attributes such as standardization, redundancy, modularity, on-board test, etc., have been designed into the system at the start of the design as opposed to incorporation at a later stage.

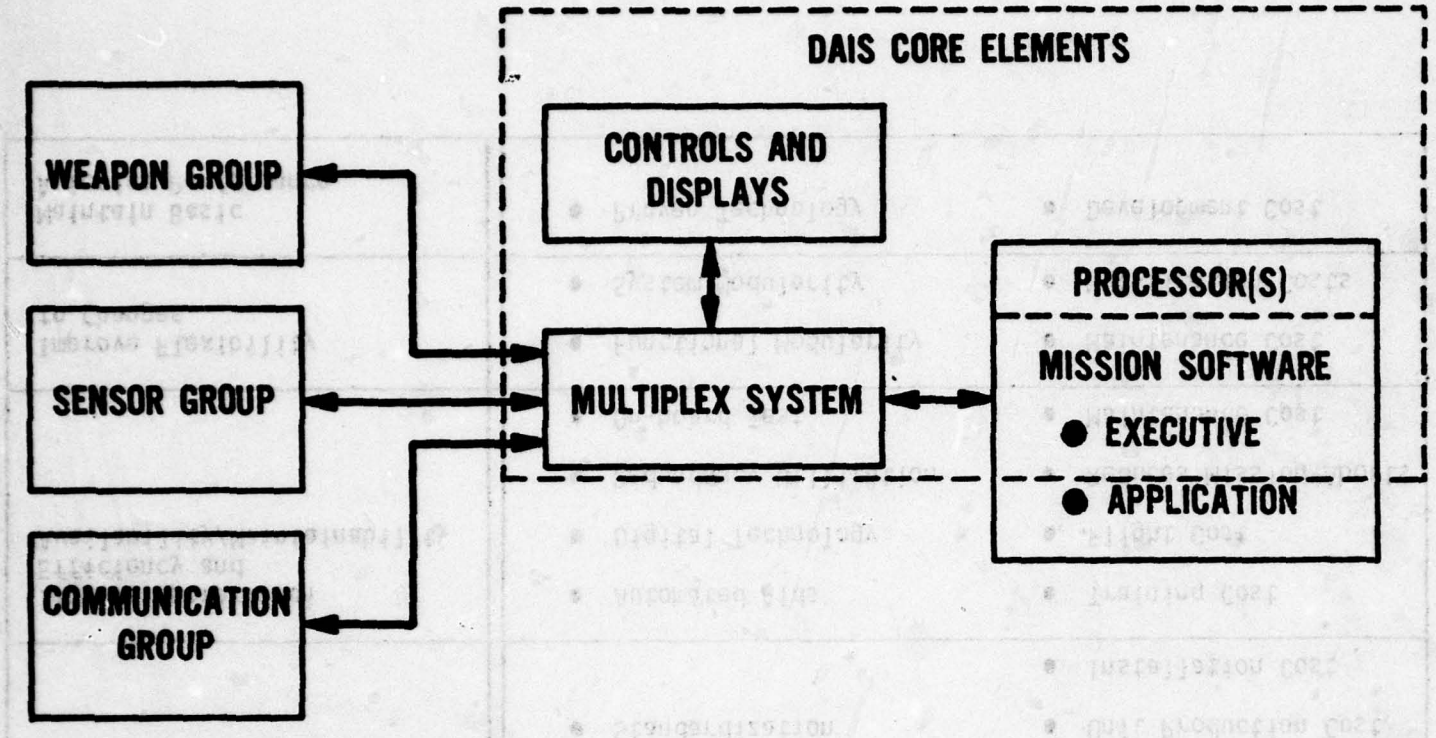


FIGURE 1. DAIS FUNCTIONAL DIAGRAM

TABLE 1. DESIGN CONSIDERATIONS UTILIZED IN MEETING DAIS OBJECTIVES

DAIS Objectives	Design Considerations	Cost Impact
Reduce Unnecessary Development Proliferation	• Replication (Core Elements) • Standardization	• Development Cost • Unit Production Cost • Installation Cost
Improve Operational Efficiency and Availability/Maintainability	• Automated Aids • Digital Technology • Redundancy Utilization • On-board Test	• Training Cost • Flight Cost • Reduces Mission Aborts • Maintenance Cost
Improve Flexibility to Changes	• Functional Modularity • System Modularity	• Maintenance Cost • Modification Costs
Maintain Basic Avionics Performance	• Proven Technology	• Development Cost

3.1 System Configuration

A block diagram of the DAIS architecture and core elements or building blocks is shown in Figure 2. These hardware core elements are the processors (with associated memory), the multiplex system (Bus Controllers, Remote Terminals, and Data Bus), and the controls and displays (pilot interface). Additional core elements are the mission software or Operational Flight Program (OFP), and the Operational Test Program (OTP). There is also a set of non-real time support software tools, namely the JOVIAL (J73/I) Compiler, the Software Design and Verification System (SDVS), and the Partitioning Analyzing and Linkage Editing Facility (PALEFAC).

The DAIS architecture as shown in Figure 2 consists of processors that communicate with each other and with the sensors, weapons, controls, and displays through a dual redundant standardized (MIL-STD-1553A) multiplex data bus system under control of mission software. The mission software as shown in Figure 2 consists of application software, which performs the processing required for a specific aircraft mission application and sensor/subsystems, and the executive software, which performs the system control and provides services to the application software. The executive software consists of the master executive, local executive, and backup master executive (if required). The master executive is responsible for the system control and resides in the processor designated the master processor. The local executive is located in each processor and provides the real time services, including data read and write, and task control to the application software. The master executive can also reside in a processor designated monitor, and provides a backup to the master executive. The executives are table driven to provide flexibility to accommodate various avionic configurations. The mission software is implemented in the JOVIAL J73/I higher order language utilizing structured programming techniques, standards, and modular architecture approach to provide reliable, reusable, and maintainable mission software.

Centralized single point system control is performed by only one processor, at any one time, and its associated Bus Control Interface Unit (BCIU), which is designated the master (Figure 2). In this control mode, the master processor/bus controller, under control of the master executive, issues commands to other devices on the selected data bus, participates in data transfers on the bus if required, checks status response from the addressed devices and interprets anomalies for all bus traffic. Remote mode is the operational state assigned to those processors and bus controllers which are not directed to be master. Remote mode functions include monitoring of the multiplex bus for commands directed to that address, responding with an appropriate status word, and storing or fetching bus data (if required). Terminals respond with the identical protocol as remote processors/BCIUs. If required for system redundancy, a processor/BCIU designated the monitor can be included in the architecture. The monitor processor, during normal system operation, responds as in the remote mode. The monitor contains the backup master executive, which is invoked during system failure modes that require backup operation.

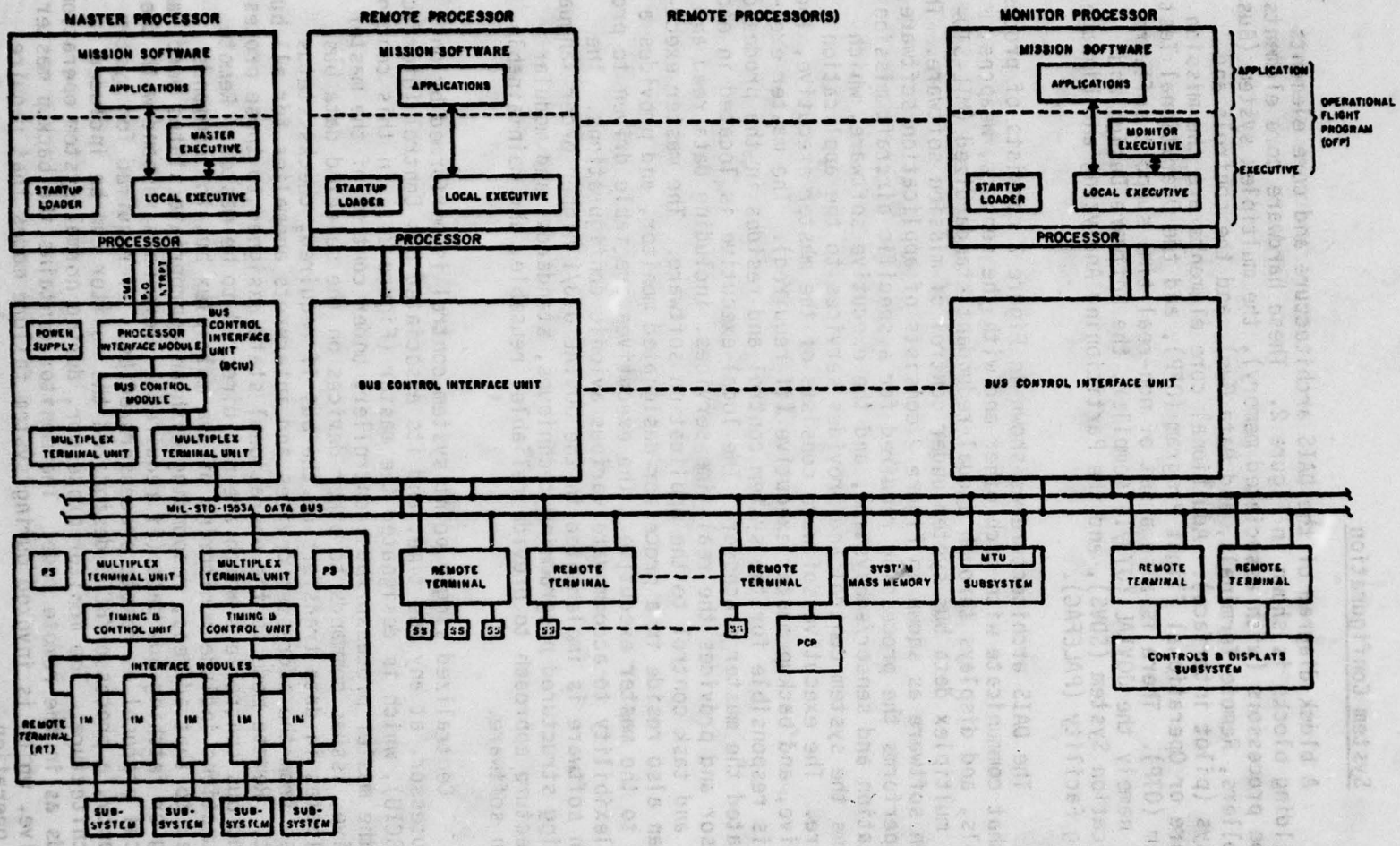
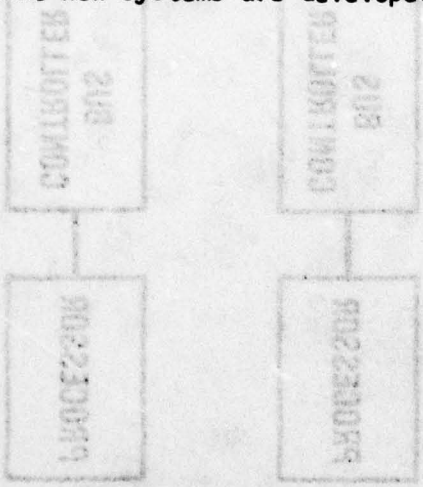


FIGURE 2. DAIS SYSTEM ARCHITECTURE

The Remote Terminals (RT) conditions via the interface modules various analog, digital and discrete signals from the sensors/subsystems and formats these for bus transmission. The RT is designed to accommodate, interchangeably, various types of interface modules to provide the proper electrical interface with different sensors, and is programmable to permit mapping of the data between the data bus and the sensors as required for the specific avionic system configuration. The avionic sensors or subsystems can interface directly with the data bus if the subsystem is compatible with the bus control protocol.

The DAIS system provides a set of application or mission independent core elements or building blocks. For a specific aircraft application, the system designer is at liberty to design or configure these building blocks to specific hardware configuration, as shown in Figure 3, and software partitioning based upon avionic subsystems and mission requirements including reliability and availability. The support software: J73/I Compiler, SDVS, and PALEFAC, provides the system designer and application programmer the tools to develop, test, and partition the application software and automatically generate the executive tables for the specific aircraft configuration.

It should be clearly understood that the basic DAIS architecture is designed for a broad class of applications and missions. The number of processors can be reduced or enlarged depending upon the avionics and mission requirements. Application independent executive software allows adaptability of this architecture to a broad class of different applications as well as to making mission-to-mission changes in a particular aircraft. Sensors, weapons, and other subsystems are selected as required for the particular mission and connected to the standard interface of the remote terminal units of the multiplex system or connected directly to the data bus. Application modules of the software will be selected or developed as required by the subsystems. This basic DAIS architecture will reduce unnecessary development duplication of similar tasks every time new systems are developed.



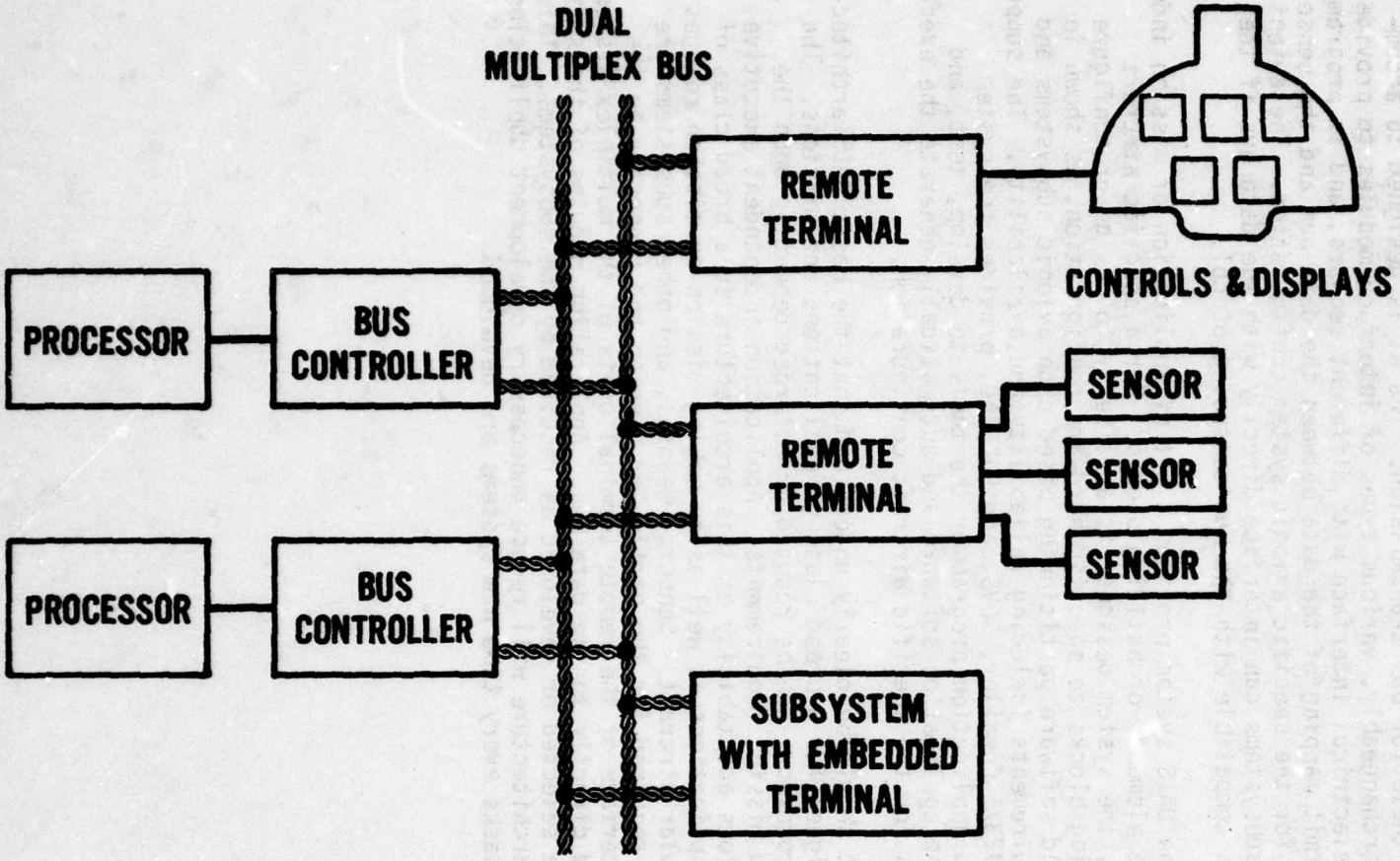


FIGURE 3. A SYSTEM CONFIGURATION

3.2 DAIS Core Elements

3.2.1 DAIS Multiplex

The DAIS multiplex provides information transfer between the elements within the system, including DAIS processors, controls and displays, and subsystems. It consists of the Bus Controller Interface Unit (BCIU) or integrated processor/bus controller, Remote Terminal units (RT), and the multiplex cable assembly (data bus). The message structure, data format, and command-response protocol is as defined in MIL-STD-1553A. The message format consists of three types of operations as shown in Figure 4 and word formats as shown in Figure 5. The system is a time division multiplex (TDM) system and a command-response system with one processor/BCIU controlling the bus traffic at any given time. Each multiplex cable assembly consists of a twisted, shielded wire pair and bus couplers.

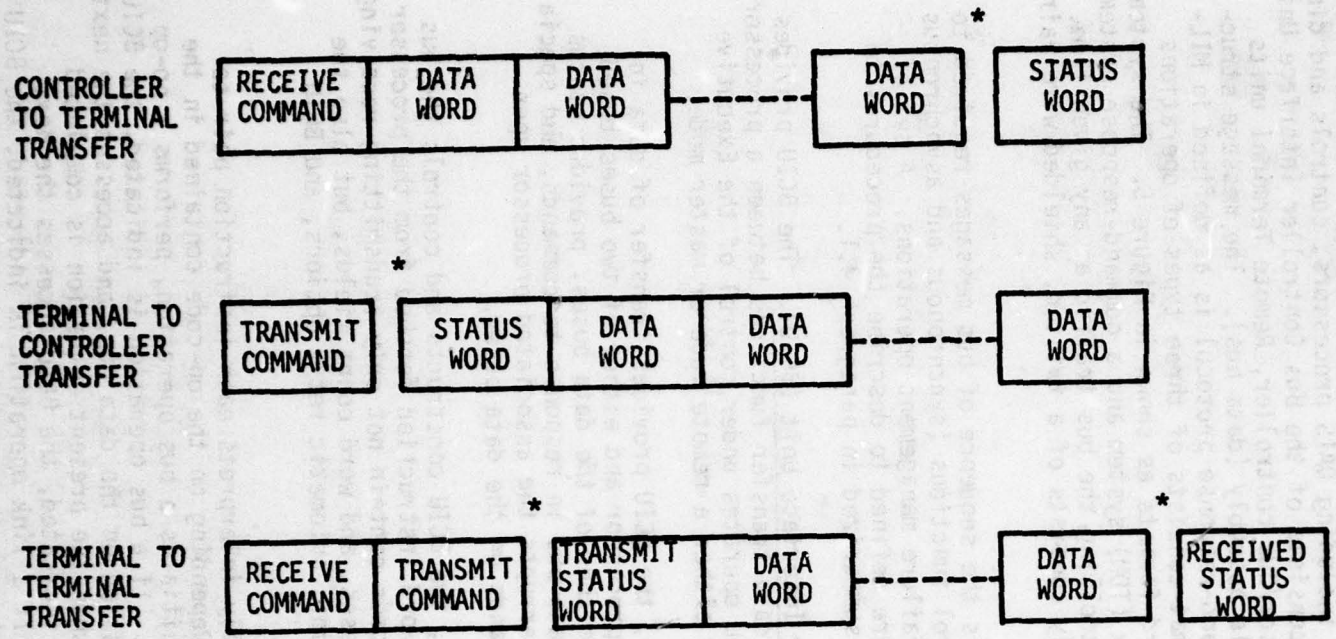
Message protocol is the sequence of bus messages required to perform system and bus control functions (synchronous and asynchronous operations), and error and failure management operations. A set of system control procedures were defined to describe the procedures to perform these operations as summarized in paragraph 4.1.

a. Bus Controller Interface Unit (BCIU). The BCIU provides the interface, control, and data transfer functions between a processor and two data buses. The BCIU operates under control of the Executive software in the processor in either a remote mode or master mode.

In the remote mode, the BCIU provides transfer of data in both directions between the processor and either of two buses based upon commands received from either of the data buses, provides status replies on the appropriate data bus in response to commands, and special internal operations, and interrupts the associated processor upon receipt of certain mode commands on the data buses.

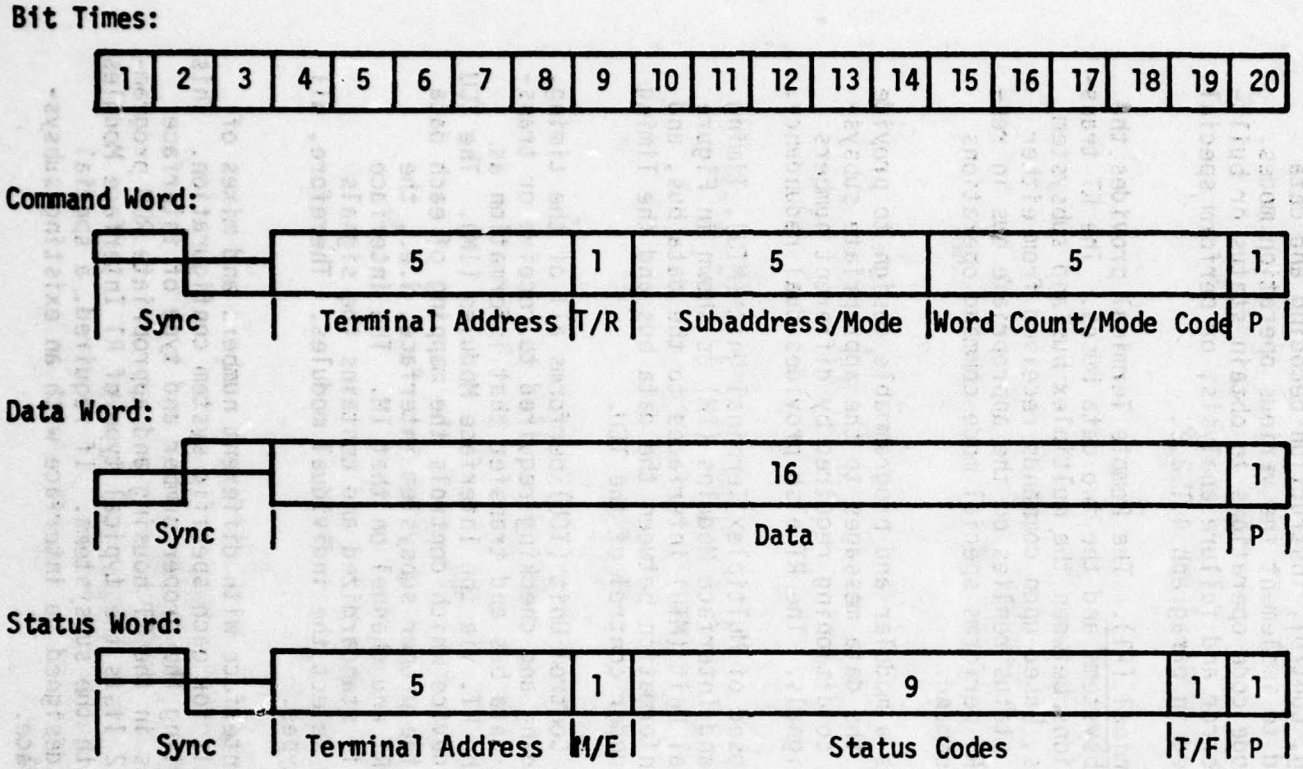
In the master mode, the BCIU constructs and controls the bus commands according to a two-word instruction fetched from the processor memory. These instruction words contain not only transmitting-receiving terminal addresses, subaddresses, and word count fields, but also the fields to dictate bus selection, automatic retry options, and BCIU internal operation codes.

The BCIU sequentially interprets each instruction pair to determine action required. Depending on the op-code contained in the instruction pair, the BCIU initiates a bus operation, performs a no-op or performs a link operation. If a bus operation is indicated, the BCIU performs the message transmission on the data bus, and accesses the next sequential instruction pair when the present operation is completed successfully. If a no-op is indicated, the BCIU accesses the next requested instruction pair. If a link operation is indicated, the BCIU performs no bus operation, but uses the second word of the instruction as the address of the next instruction pair.



*Gap Time

FIGURE 4. MESSAGE FORMATS



P - Parity
T/R - Transmit/Receive
M/E - Message Error
T/F - Terminal Failure

FIGURE 5. WORD FORMATS

Figure 6 identifies the major components that comprise a BCIU. Each Multiplex Terminal Unit (MTU) provides the interface function to one data bus. The Processor Interface Module (PIM) provides the interface function to the processor. The changing of processor types is accommodated by a re-design of only the PIM. The Bus Control Module (BCM) provides the timing, control, instruction decoding and data transfer routing required to implement the various operation modes. The BCIU also performs mode code operations to obtain status or built-in-test information for error and failure analysis, or perform special mode operations as defined in paragraph 4.1.2.2.

b. Remote Terminal (RT). The Remote Terminal provides the interface between the subsystems and the two data buses. The RT transfers data in both directions between the multiplex bus and subsystem via the Interface Modules, based upon commands received from either data buses, and provides status replies on the appropriate bus in response to commands. The RT performs special mode command operations upon receipt of a mode command.

The RT provides a modular and programmable design to provide flexible partitioning of the data messages to the appropriate subsystems/sensors, and signal conditioning required by different numbers and mixes of subsystem signals. The RT also provides dual redundancy.

The RT is composed of Multiplex Terminal Unit (MTU), Timing and Control Unit (TCU), and Interface Modules (IM) as shown in Figure 7. The Multiplex Terminal Unit (MTU) interfaces to the data bus, and transmits and receives information between the data bus and the Timing and Control Unit (TCU), under control of the TCU.

The Timing and Control Unit (TCU) performs all of the timing, control, buffering, decoding and checking required to receive or transmit information from the data bus and transfer that information as outputs or inputs from the RT, via the Interface Modules (IM). The TCU contains a programmable device which controls the mapping of each data word in each message to the proper subsystem interface, i.e., the specific IM slot in the RT and channel on that IM. The interface between the TCU and IMs is standardized and contains the signals required to allow TCU to select the individual modules. Therefore, all IM slots accept all IM types.

Each RT will interface with different numbers and mixes of subsystems/sensors signals for each specific system configuration. This is accomplished by inserting the proper number and type of interface modules into the IM slots in the RT housing and appropriate ROM programming in the TCU. Table 2 lists the typical type of RT Interface Modules provided to interface with the subsystems. If required, a special interface module can be designed to interface with an existing subsystem with a unique interface.

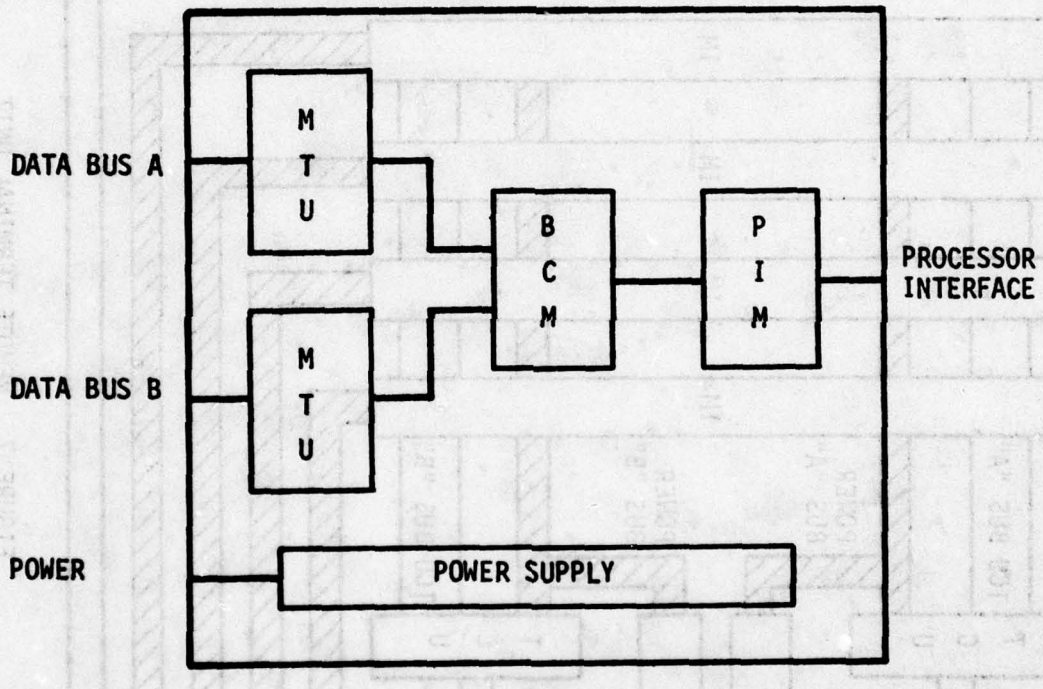


FIGURE 6. BUS CONTROL INTERFACE UNIT

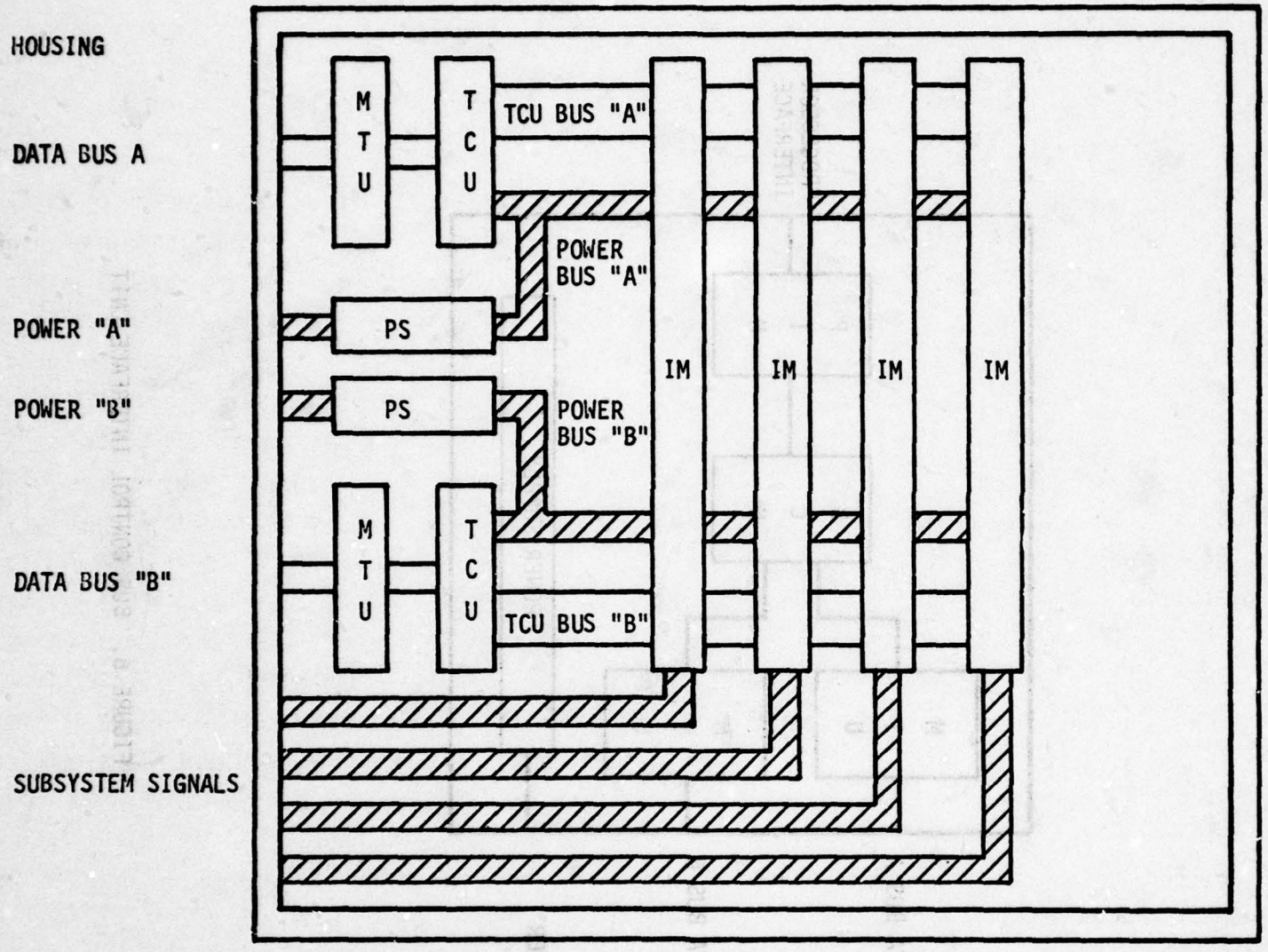


FIGURE 7. REMOTE TERMINAL UNIT

TABLE 2. RT INTERFACE MODULES

IM Type	Channels Per IM	Inputs/Outputs Per Channel	Total Inputs/Outputs Per IM
Differential Discrete Input Module	1	16	16
Differential Discrete Output Module	1	16	16
Single Ended Discrete Input Module	2	16	32
Single Ended Discrete Output Module	2	16	32
Momentary Closure Input Module	1	16	16
Switch Closure Discrete Input Module	1	16	16
Switch Closure Discrete Output Module	1	16	16
A.C. Analog Input Module	8	1	8
A.C. Analog Output Module	8	1	8
D.C. Analog Input Module	8	1	8
D.C. Analog Output Module	4	1	4
Facility Input/Output Module	1	1	1
Synchro Input Module	8	1	8
Synchro Output Module	8	1	8
Serial Digital Input	4	Up to 32 words	
Serial Digital Output	4	Up to 32 words	
Pulse Counter-Torques Module	1	3	3

The MTU & TCU functions of the RT can be embedded in a sensor or subsystem, so that the interface of the sensor or subsystem is directly with the multiplex data bus. Functionally the embedded RT would respond to commands received on either data buses the same as an RT.

3.2.2 DAIS Processors

The DAIS processors (AN/AYK-15) provide computational and control, memory and input/output capabilities. The DAIS processors are distributed in the architecture, interconnected through the DAIS multiplex system. In this architecture, the processors address only their own memory units.

The DAIS processors include the following features:

- Central Processor Unit (CPU)
  - 16 general registers
  - Extensive instruction repertoire
  - 379K operations per second throughput based upon a specified benchmark program
  - Floating point capability
  - Direct, indirect, immediate and index addressing modes
  - Programmable interval timers
  - Vectored interrupts
- Memory
  - Exapndable random access memory to 65K words (16 bit words), in 16K word memory modules
  - Memory parity and write protect features
- Input and Output
  - Discretes
  - Program controlled
  - Direct Memory Access (DMA)
  - External interrupts

The integrated DAIS processor/bus controller combines the processor and BCIU functions into an integrated unit and incorporates the MIL-STD-1750 instruction set. This integrated DAIS processor/bus controller (AN/AYK-15A) includes the following features:

- Processor
  - 16 general registers
  - Extensive instruction repertoire as specified in MIL-STD-1750
  - 379K operations per second throughput based upon specified benchmark program
  - Floating point capability
  - Direct, indirect, immediate, base relative, IC relative, and index addressing modes
  - Programmable interval timers
  - Vectored interrupts
- Memory
  - Expandable random access memory to 65K words (16 bits per word), in 16K word memory modules
  - Memory parity and write protect features
- Input and Output
  - Discretes
  - Program controlled
  - Direct memory access
  - External interrupts
  - Multiplex data bus } Added feature to -15A
  - Bus controller }

3.2.3 DAIS Controls and Displays

The DAIS controls and displays consist of a set of shared multipurpose data entry devices, control devices, and display devices to interface the pilot with various avionic system configuration. The DAIS Controls and Displays concept is based upon an integrated set of common and shared devices which can be used for most of the avionic functions. Thus, the DAIS control and display concept embodies the following features:

- Flexibility to accommodate changes in avionic functions and moding as a result of new sensors/subsystems or mission changes without requiring a physical change in the control and display devices.

- Efficient design for centralized system management by the pilot as required for the system application.

- Redundancy if required for specific system application (e.g. CRT's serve as backup to each other, redundant display generators) for backup of mission critical functions.

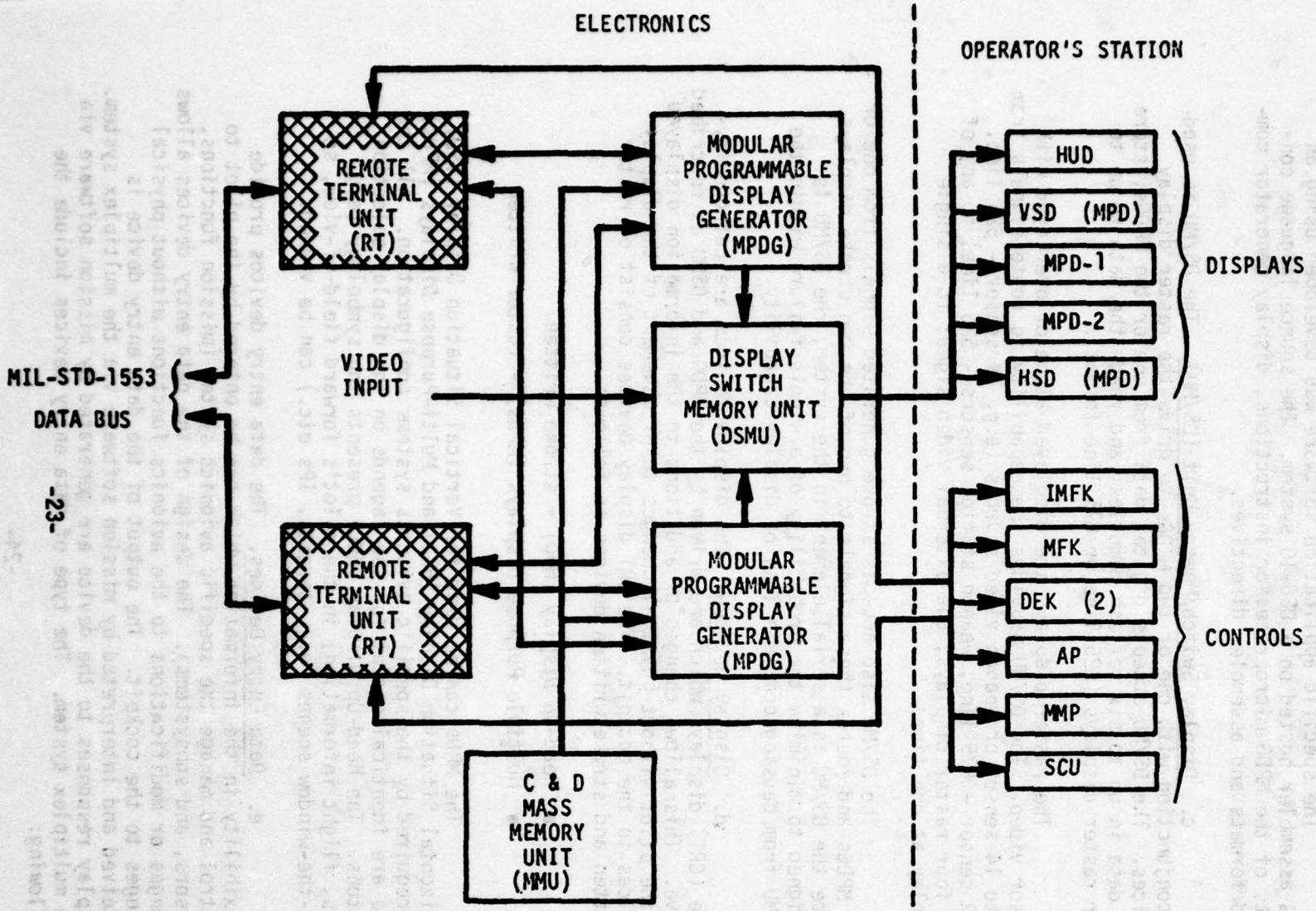
- Provide for mode control and command data from data entry devices and to display devices to be transmitted and received over the multiplex bus from mission software.

The set of control and display equipment developed for the DAIS demonstration is a representation of the architecture one would utilize in the DAIS architecture. An example of the controls and displays subsystem configuration is shown in Figure 8. The units or building blocks of the DAIS controls and displays include the following:

a. Modular Programmable Display Generator (MPDG). The MPDG is a programmable display graphic generator, capable of being programmed to generate graphics composed of predefined symbols, line segments, and alphanumeric symbols. The MPDG generates symbols for presentation on both raster and stroke written display devices. The MPDG generates the display data for up to four raster display devices via the Display Switch/Memory Unit (DS/MU) which contains the refresh memory for the raster displays. The MPDG also generates stroke (x and y deflection signals, and blank/unblank signals) for one stroke display.

The MPDG generates various forms of symbology including geographic map symbols, overlay symbols, tactical horizontal and vertical situation data, flight situation and director symbols as programmed in the C/D subsystem application software. This application software executes on the programmable processor in the MPDG.

b. C/D Subsystem Application Software. The C/D subsystem application software consists of the routines and display lists for the display generator in the MPDG to generate the display symbology, text formats and pages, map data, etc. required for a specific avionics system application. Mode control and data messages received from mission software via the multiplex system controls the display modes, display assignments, text displays, and variable symbology. The application software as commanded by mission software performs the loading of the MPDG programs, text pages, and map data from the C&D Mass Memory Unit (MMU).


ELECTRONICS
OPERATOR'S STATION
REMOTE TERMINAL UNIT (RT)
MODULAR PROGRAMMABLE DISPLAY GENERATOR (MPDG)
HUD
VSD (MPD)
MPD-1
MPD-2
HSD (MPD)
DISPLAYS
MIL-STD-1553 DATA BUS
VIDEO INPUT
DISPLAY SWITCH MEMORY UNIT (DSMU)
IMFK
MFK
DEK (2)
AP
MMP
SCU
CONTROLS
REMOTE TERMINAL UNIT (RT)
MODULAR PROGRAMMABLE DISPLAY GENERATOR (MPDG)
C & D MASS MEMORY UNIT (MMU)

The specific application software was developed using an MPDG assembler hosted on a DEC-10 system. The source language consists of the MPDG microprocessor instructions, display generator command formats and assembler directives.

c. Display Switch/Memory Unit (DS/MU). The DS/MU is used in conjunction with one or two MPDGs to drive the raster display devices. The DS/MU accepts digital data from one or two MPDGs; store the data in up to six refresh memories; and reads the data to up to four raster displays (525 line or 875 line format).

The raster displays are displayed alone or overlayed with sensor video. The DS/MU provides the capability to route signals from up to 14 sensor or weapon video sources (e.g. TV sensor - 525 line, FLIR sensor - 875 line, weapon station sensors - 525 line) to any of the four raster displays, one external video signal to a single display at one time.

The DS/MU also accepts stroke generated signals from one or two MPDGs and routes these signals to one of the two stroke displays. Since the DS/MU is a serial element in the system, the DS/MU is designed to minimize the probability of a single failure within the DS/MU from causing complete loss of display capability.

d. Display Devices. The display devices are cathode-ray-tube (CRT) displays which are driven by the MPDG and DSMU as described above. This allows changes or additions to the information displayed to the pilot without physical changes to or movement of the display devices in the cockpit. The CRT display devices consist of raster written and stroke written devices as follows:

- Head-Up Display (HUD) - stroke written
- Multiple Purpose Displays (MPDs) - raster written

The MPDs can be used for Vertical Situation Display (VSD), Horizontal Situation Display (HSD) and Multi-Purpose Displays (MPDs) as required by the specific avionics systems configuration. All the MPDs are identical except for the legends on the display selection buttons. The Head-Up Display (HUD) presents the symbology (e.g. cues, flight information) in the pilot's forward field-of-view, so out-the-window scenes (e.g. targets, IPs etc.) can be viewed.

e. Data Entry Devices. The data entry devices provide flexibility in the information which can be entered by the pilot to control and manage the specific avionics system (mission functions, sensors, and subsystems). The design of the data entry devices allows changes or modifications to the avionics functions without physical changes to the cockpit. The output of the data entry device is received and interpreted by mission software via the multiplex system. Display responses to the device are generated by mission software via the multiplex system. The type of data entry devices include the following:

1. Integrated Multifunction Keyboard (IMFK). The IMFK consists of a programmable alphanumeric 280 character display, with seven master function pushbuttons, and ten submode select pushbutton switches, five on either side of the display. The IMFK display is capable of displaying ten rows of characters divided into three columns (nine characters in left and right columns, and ten characters in center column). The display information is generated by mission software. The pushbuttons generate a request via the remote terminal for the master executive to asynchronously service and transmit the data to the appropriate processor. Application software interprets the pushbutton or switch action and provides the appropriate response.

2. Multifunction Keyboard (MFK). The MFK consists of seven master function pushbuttons, ten submode pushbutton switches, each with a rear projection display capability, and ten indicator lights. The MFK interfaces with mission software via the RT like the IMFK, and can be utilized for specific control and display functions or as a backup for the IMFK as required for the specific mission applications.

3. Data Entry Keyboard (DEK). The DEK provides the pilot with a means of entering numeric and limited alpha data into mission software via the multiplex data bus. The DEK stores and displays up to ten characters. When the enter key is depressed, a request for services via the RT is made to the master executive. Upon receiving the data, mission software application software interprets the data depending upon the selected submode via the IMFK or MFK.

f. Control Devices. The Control and Display subsystem also provides several control devices to manage processor and multiplex power, and system control as follows:

1. Processor Control Panel (PCP). The PCP provides switches for signaling control of power to the multiplex system (RT sides A & B) and to each processor/BCIU. Application of power to each processor/BCIU initiates the startup sequence. The PCP also includes momentary switches to enable the startup (cold start) and restart (warm start) of the system.

2. Sensor Control Unit (SCU). The SCU provides for selection and position control of a symbol or sensor (X and Y directions) vis mission software. The hand controller is a force control with DC voltage output which is proportional to the force in the X and Y directions. The SCU also provides trigger and switch discrete outputs. These outputs are transmitted to mission software via the multiplex data bus; and mission software drives the appropriate symbol or sensor based upon the selected mode. The SCU also provides a control knob to control cursor brightness.

3. Armament Panel (AP). The armament panel provides switches for weapon armament control (bombing, fusing, gun firing, and jettison).

4. Master Mode Panel (MMP). The MMP consists of fifteen master mode switches, each with an indicator light and HUD related switches & controls. When a master mode switch is depressed, a request for service via the RT will be made to the master executive. Upon receiving the data, mission software interprets and responds to the selected master mode.

The interface between the pilot and the integrated set of controls and displays is dependent upon the specific avionic mission and system configuration. The system designer would select the set of integrated controls and displays required for the specific avionic system, and define the pilot sequences and display responses required to control and mode the avionics system.

The interface between the mission software and the C&D application software in the MPDG is also mission and application dependent. This interface is defined by the data bus messages from mission software to the MPDG to perform the following functions:

- Load the MPDG
- Control display (VSD, MPD, HAD, & HUD) assignment
- Control display modes
- Data to drive the display symbology, e.g. elevation, altitude, heading, etc.
- Text data

With increased avionic and mission complexity, the role of the pilot is changing to that of a total system manager. Through the mission software, a systems management function can be performed to collect information and control a functionally integrated set of subsystems and sensors, thus reducing the pilot workload. Functions such as subsystem checkout and monitoring, data computation and reduction, routine sequencing within mission modes, mission and aircraft data storage can be performed by the mission software during normal operating modes or master modes with minimum pilot intervention. Operation by exception can be the design into the system control and operation.

For example, the pilot can use the integrated controls and displays to mode the avionics system through the mission software application functions. He receives displayed information from the software about the mission and avionics system operations. The control devices permit the pilot to establish the operations to be performed by the

software functions. The master modes set the display devices to prescribed formats via mission software and the MPDG. The pilot, however, can override these normal operations to either cause alternate modes or activate operations not required by the selected master mode.

Master modes can be defined for specific phases of the mission where certain normal operations are active. Upon entering a new master mode, some previous operations can be terminated as new ones are activated. Manual modes previously selected by the pilot can be retained if compatible with the new master mode. If the new master mode requires a system that has been deselected using the manual mode, the pilot can be alerted through displayed exception messages. The master mode panel, as labeled for the specific mission applications, permits the pilot to select these prescribed master modes. Mission software, in response to the selected master mode, will command the C/D application software to generate prescribed formats for the various displays (HUD, VSD, MPD, etc.). The pilot can override the preprogrammed mission functions through the IMFK displayed pages and side keys which are also handled by mission software. The pilot will also be able to enter information into mission software via the DEK. For the DAIS Demonstration, a representative set of master modes and mission operational sequences were defined and implemented under control of mission software application software and C&D application software. The requirements for these modes were defined in a Pilot/Operational Sequence Interface Control Document (ICD).

3.2.4 DAIS Mission Software

The mission software resides in the DAIS processor(s) and is separated into the executive software and the application software. The mission software consists of the Operational Flight Program (OFP) and the Operational Test Program (OTP). The OFP application software performs the specific avionic mission functions, while the OTP application software is used on the ground, on board the aircraft, to test and isolate failures to the LRU level.

3.2.4.1 Executive

The Executive is organized into the local executive and master executive.

a. Local Executive. Each DAIS processor contains its own local executive. The local executive software is table driven and performs the following minimum functions:

1. Process Control - provides services for the application software to activate and deactivate periodic and non-periodic tasks when appropriate conditions have been met. These conditions are based upon logical settings (on or off) of real time events.

2. Event Control - provides the mechanism for setting, resetting, and evaluating real time events which communicate conditions (on or off) signaled between tasks whether in the same or different processors.

3. Data Control - guarantee integrity of shared data and provide mechanism for transmission and reception of data over the multiplex bus.

4. Processor initialization - provides initialization for processor power transient recovery, initializes program loads, and minor cycle synchronization with the other processors.

b. Master Executive. The master executive software is also table driven and manages the system configuration by performing the following minimum functions:

1. System Synchronization Control - allocates time segments (minor cycle and major cycle) for synchronous messages and performs minor cycle synchronization by transmitting minor cycle master function mode commands to the other processors.

2. Bus Control - controls transmission of synchronous and asynchronous messages over the multiplex data bus.

3. System Error/Failure Management - monitors and analyzes errors and failures based upon both bus and terminal devices. Initiates message retry procedures to recover from message errors.

4. Configuration Management - detects and isolates permanent device failures, maintains system configuration status, reports failure to the application software, and initiates backup or recovery operation if required.

5. Mass Memory Management - provides for retrieving and storing information from the mass memory on request.

3.2.4.2 Mission Software Architecture

The structure imposed upon the mission software is general enough to accommodate mission-to-mission changes or changes in the complement of avionic sensors. It allows transition of the application independent executive software as well as valid application tasks to other aircraft configurations using the DAIS architecture (system modularity).

The application software is organized in a hierarchial control tree structure. All application software consists of tasks which can be either controllers or calculators. Each task performs a specific function based upon its inputs, and required for its outputs. Control over the tasks or modules conforms to this hierarchial control tree structure. This structure not only defines the control structures, but shall also facilitate modular application software for maintenance and growth. The application software would consist of the tasks to control the various mission dependent sensors (e.g., inertial navigation system, instrument landing system, TACAN, radar altimeter, air data, etc.) and subsystems (e.g., stores, etc.); and perform the various mission functions (e.g., navigation, weapon delivery, steering navigation update, acquisition/cueing, target fix, stores management, communication, etc.).

The structures described above are further divided into modular components. Each component of the computer program is a closed program, i.e., single entry and single exit point. In addition, each component is part of a hierarchial structure where each component is controlled by a component of the next higher level of the hierarchy. The components are chosen that isolate hardware functions, to localize the impact of changes to the equipment, and allow for mission-to-mission system changes.

Structured programming techniques are used as a tool to increase reliability, understanding, and to eliminate intricate logic that is difficult to validate and verify. Only closed logic structures --logic structures which have a single entry point and a single exit point--are employed in the construction of program components. To further emphasize standardization and understanding, the higher order language - JOVIAL J73 - is used for development of all the mission software except for those portions of the program where computer time must be minimized or a high degree of numerical accuracy is required or interface with processor or bus controller hardware.

3.2.4.3 Application Software

The application software of the OFP is structured to facilitate maintenance and growth. The functional categories are specified in the following:

a. Master Sequencer. The master sequencer is at the top of the application software hierarchy and controls the request processor, configurator, and subsystem status monitor. This master sequencer is only required in the master and monitor processors.

b. Request Processor. The request processor responds to pilot's requests and invokes the appropriate application functions that generate information for displays or for controlling avionic support subsystems.

c. Configurator. The configurator:

1. Defines the set of application functions to be invoked by request from the request processor or the subsystem status monitor.

2. Generates a new set of available functions upon significant changes in mission phasing or equipment moding. This set of functions is based on a global knowledge of overall system status.

d. Subsystem Status Monitor. The subsystem status monitor monitors subsystem failures and communicates the information to the configurator.

e. Mission Operations. The mission operations (OPS):

1. Acts as a control authority for operations either when invoked by phase selection by the pilot or automatically when current phases are terminated in an emergency or abnormal condition.

2. Exists, as a minimum, for preflight/inflight startup, take-off, enroute, combat modes, approach, landing, and postflight shutdown operations.

f. Specialist Functions. The specialist functions (SPECS) provide functions required by OPS modules or by the pilot (e.g., navigation, onboard test, weapon stores, etc.).

g. Equipment Modes. The equipment modes (EQUIP) provide the interface for each piece of avionics equipment (inertial, communication equipment, etc.).

h. Display Processors. The display processors (DISP) provide specific display messages to drive the symbols and graphical displays on the DAIS controls and displays.

3.3 Other Elements

3.3.1 System Mass Memory

This optional subsystem supports the DAIS capabilities to load/reload program modules and mission dependent data, and to record data for post flight analysis.

The processor/BCIU in control of the DAIS multiplex data bus manages the system mass memory operation. This is the control processor during startup, the master during normal operation, or the monitor during backup. All error handling as well as the translation of file level information to device level information is performed by this processor.

A fixed RT address is required for the system mass memory in order for it to be accessed by the bootstrap routine during startup. Four subaddresses are required during normal operations to support the following basic functions:

Address Select - to set a starting address for the mass memory and specify if synchronous or asynchronous operations are to be performed. Also to set read mode or set write mode.

Address Retrieve - to obtain the current (i.e. next to be accessed) mass memory address and settings of the Read/Write and Sync/Async flags.

Read - to transfer data from the mass memory and advance the current address by the number of words transferred.

Write - to transfer data to the mass memory and advance the current address by the number of words transferred.

The system mass memory is word addressable and contains approximately a million 16-bit words. It is self-contained in that a directory on the system mass memory identifies the location of all files stored in the memory. A basic loader for the processor is stored at a fixed location on the system mass memory, and therefore, available to be read by the startup bootstrap in the processor.

The system mass memory includes a buffered interface to minimize access delays. When an access delay is encountered, this is indicated by a "not ready" bit in the status word returned on the bus.

3.4 Non-Real Time Support Software

The DAIS non-real time support software provides the tools to develop, generate, test and partition the mission software for the specific avionics configuration.

3.4.1 JOVIAL J73/I Compiler

The JOVIAL compiler is a non-real time program which translates high-order-language statements (J73/I source code) into software object modules (mission software and support software) which can be loaded and executed on specified target computers (e.g. DAIS processors, DEC-10 computer). The JOVIAL language is machine independent, using self-explanatory English words and familiar notation for algebraic numerical computations and logical operations. The JOVIAL compiler provides a common, powerful, and easily understandable programming language, for a wide range of applications. The compiler provides capability for designating and manipulating (sharing) data between interdependent programs or tasks through a communication pool (compool).

The J73/I compiler is designed to be easily retargeted to produce object code for a new and unspecified computer. The J73/I compiler is also designed to be easily rehostable to another host computer. Initially, the J73/I compiler was hosted on the DEC-10 computer system, and targeted for the DAIS processor and DEC-10 system.

The structure of the J73/I compiler is shown in Figure 9. The system programmer will generate the J73/I source input and compool input files using the JOVIAL J73/I User's Guide. The programmer will then compile the source program using the J73/I compiler which shall produce source listings, cross references, object listings, and the relocatable object files for the selected target computer.

The output of the J73/I compilation is relocatable module file. In order to execute the program on the DEC-10, it must be converted from relocatable file to core image form using the DEC-10 (LINK-10) linker loader. In order to execute the program on the DAIS processor, it must also be converted to core image form using the HBC linker loader.

The J73/I User's Guide provides definition of the J73/I language, properties, and statements including examples of how to write a J73/I program. The guide also presents a syntactic description of the language elements with examples and necessary information for user to write, compile, load and execute programs on the target computers.

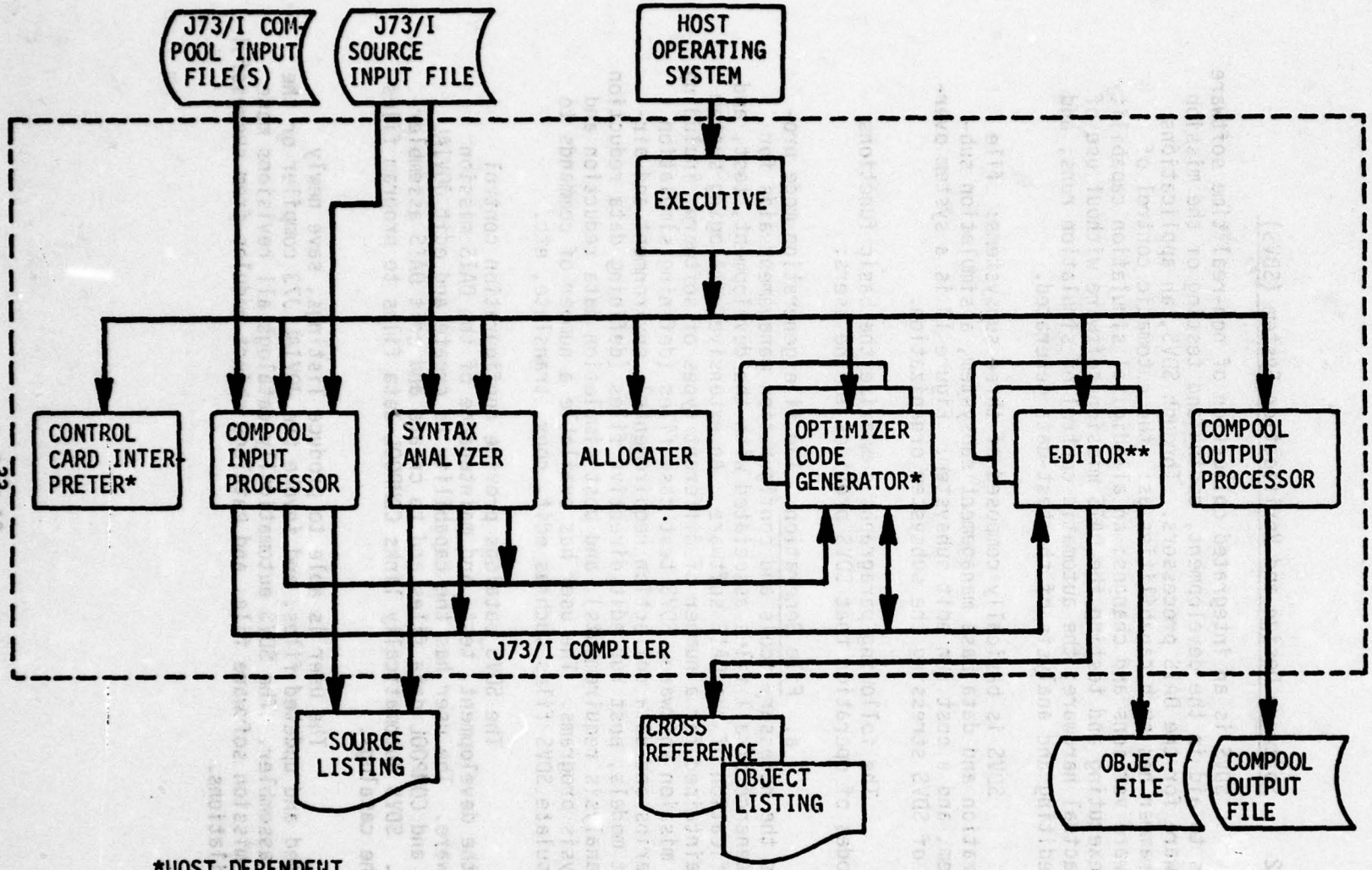


FIGURE 9. J73/I COMPILER STRUCTURE

3.4.2 Software Design and Verification System (SDVS)

SDVS is an integrated collection of non-real time software tools to aid in the development, coding and testing of the mission software for the DAIS processors. Through SDVS, an applications programmer has such capabilities as: the automatic control of software versions and changes; an all-digital simulation capability for executing and testing the DAIS mission software without use of the actual hardware; the automatic control of simulation runs; and the editing and analysis of the test-data generated.

SDVS is basically composed of three subsystems: file generation and data base management subsystem, a simulation subsystem, and a post run edit subsystem. Figure 10 is a system overview of SDVS stressing the subsystem organization.

The following paragraphs summarize the basic functions or modes of operation that SDVS provides to the users:

a. File Generation. The file generation mode provides the necessary tools and configuration management aids for maintenance of all files associated with the development, test, and verification of the DAIS software. An extensive cataloging system is maintained for a number of different types of software, including DAIS mission software, SDVS test case files (defining simulation scenarios and data collection requirements), environment and aircraft models, post run edit directive files (defining data reduction and analysis requirements), and post simulation data reduction and analysis programs. The user has available a number of commands to manipulate SDVS files such as edit, copy, translate, etc.

The SDVS catalogs provide configuration control for the development, test, and maintenance of the DAIS mission software. The user has the capability to create and edit JOVIAL code and COMPOOL data files and to create and edit DAIS assembler code. SDVS automatically links COMPOOL data files to program files in the catalogs.

The user is able to produce listings, save newly created and updated files, and invoke the JOVIAL J73 compiler or the DAIS assembler. The SDVS automatically catalogs all revisions made to a mission software file, and catalogs object modules from successful compilations.

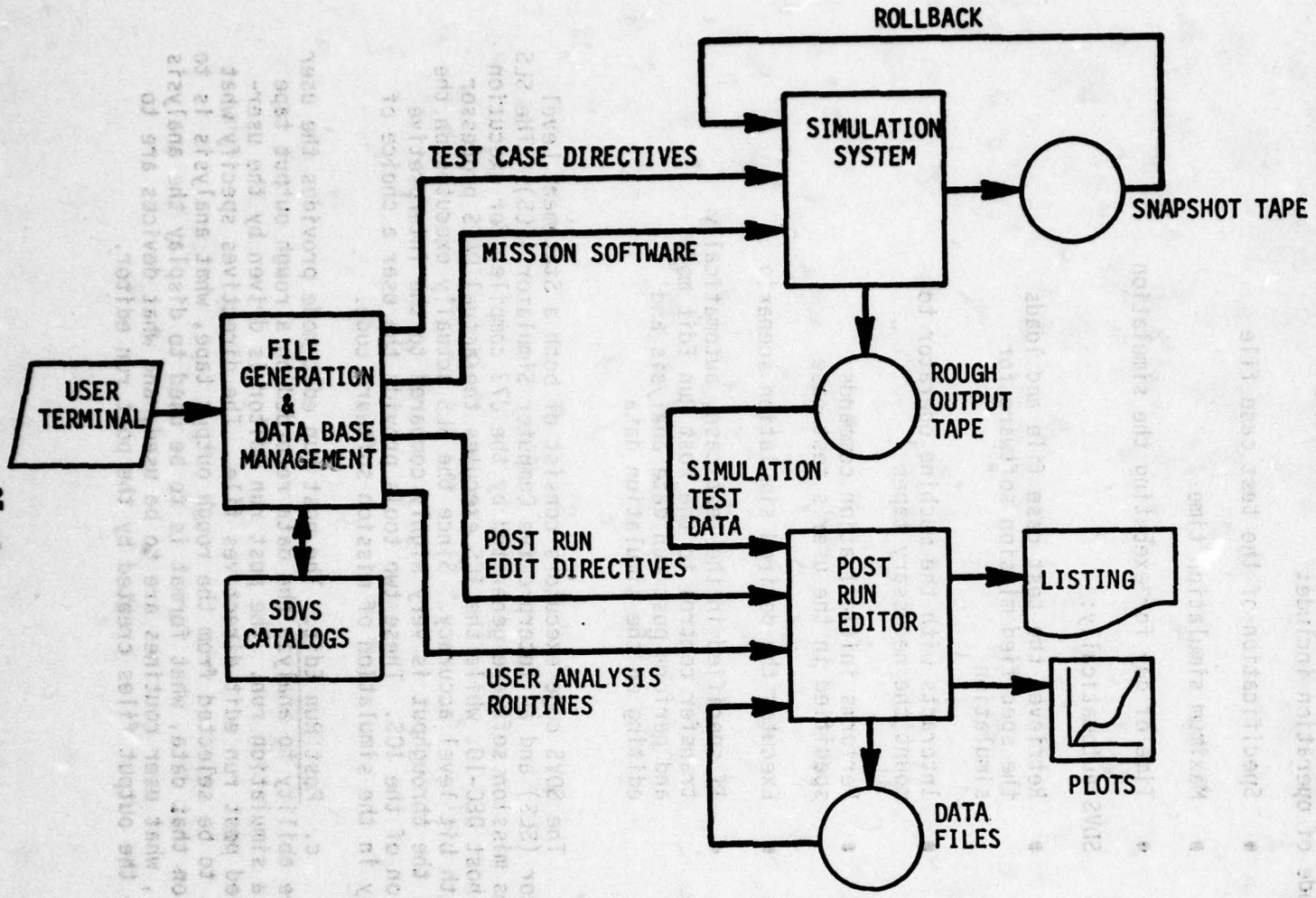


FIGURE 10. SDVS SYSTEM OVERVIEW

b. Set Up and Run Simulation. This mode of operation is used to submit a simulation run based on a test case directives file that has previously been created and translated. The user inputs for this mode of operation include:

- Specification of the test case file
- Maximum simulation time
- Time of day for executing the simulation

SDVS automatically:

- Retrieves the test case file and loads the specified mission software for simulation
- Interacts with the machine operator to mount the necessary tapes
- Performs initialization commands specified in the user's test case
- Executes the desired simulation scenario
- If specified in the test case, automatically transfer control to the Post Run Edit mode and perform post run data analysis and editing on the simulation data

The SDVS code executors consist of both a Statement Level Simulator (SLS) and an Interpretive Computer Simulator (ICS). The SLS executes mission software generated by the J73 compiler for execution on the host DEC-10, while the ICS executes the actual DAIS processor code with bit level accuracy. Since the SLS actually executes on the DEC-10, the throughput is very highly compared to the interpretive execution of the ICS. These two tools provide the user a choice of fidelity in the simulation of mission software code.

c. Post Run Edit. The post run edit mode provides the user with the ability to analyze the data recorded on a rough output tape during a simulation run. The post run editor is driven by the user-specified post run edit directives file. The directives specify what data is to be selected from the rough output tape, what analysis is to be run on that data, what format is to be used to display the analysis results, what user routines are to be used, and what devices are to receive the output files created by the post run editor.

The post run editor provides tabular printouts and data plots based on user directives. An important feature is the ability for a user to write an analysis routine in JOVIAL and have it execute within the framework of the post run editor. The post run editor mode performs in batch or on line with results displayed on the users terminal.

d. Rollback. The user may specify that all data necessary to restart a simulation, from a particular point in the simulation, be saved on a snapshot tape. The rollback function is using the snapshot tape to provide simulation restart capability. The user may use the original test case or he may modify the original directives to obtain additional output or to alter existing simulation conditions.

The output of this mode of operation is a simulation run which (if no changes were made in the test case directives) will exactly match the previous one. Changes may be made to provide further analysis of a simulation run which is of special interest.

d. Delete Mode. This mode of operation shall be only available to the SDVS manager and provides him the capability to delete files from the various SDVS catalogs. This function shall be made a manager level function to allow manager level control over the disposition of all files.

f. Supervisor Mode. This mode, like the delete mode, shall be available only to the SDVS manager for configuration control purposes. One of the features of the SDVS file management scheme is that before a user may generate any file in the file generation mode, specifications must have been created for that file, including the provision of a list of users who are authorized to write on the file, and if appropriate, a list of users who are free only to read it. The creation of file specifications are performed from supervisor mode. This mode is entered only by a SDVS user logged in on the special number.

3.4.3 Partitioning Analyzing and Linkage Editing Facility (PALEFAC)

PALEFAC is a non-real time support software tool for use by the system designer and application programmer to:

1. Build the data tables for the DAIS local and master executives.
2. Provide bus analysis and module partitioning information.
3. Produces the executive tables in J73/I source code.
4. Generate linker command files for each DAIS processor for the DEC-10 Linker or DAIS Processor Linker.

PALEFAC may be used as a standalone tool, or under control of SDVS. PALEFAC consists of three basic programs, as shown in Figure 11; PALEFAC pre-processor, PALEFAC main program, and PALEFAC module input (PMI) decoder.

PALEFAC is used to build the mission software load modules for each of the DAIS processors. Inputs to PALEFAC are the application software modules including the executive service requests generated by the application programmer in J73/I source code. The pre-processor reads each application module and creates a record for each application module in the PMI file.

The system designer will prepare PALEFAC Global Input (PGI) file based upon the specific system configuration, partitioning of application tasks to each processor, and bus messages to each terminal. This would include:

a. Bus Messages

- To/from data block name or Terminal Address/ Subaddress
- Cycle (period and phase) for synchronous transmissions
- Activity request identification for asynch-ronout transmission
- Word count
- Class for message error retry

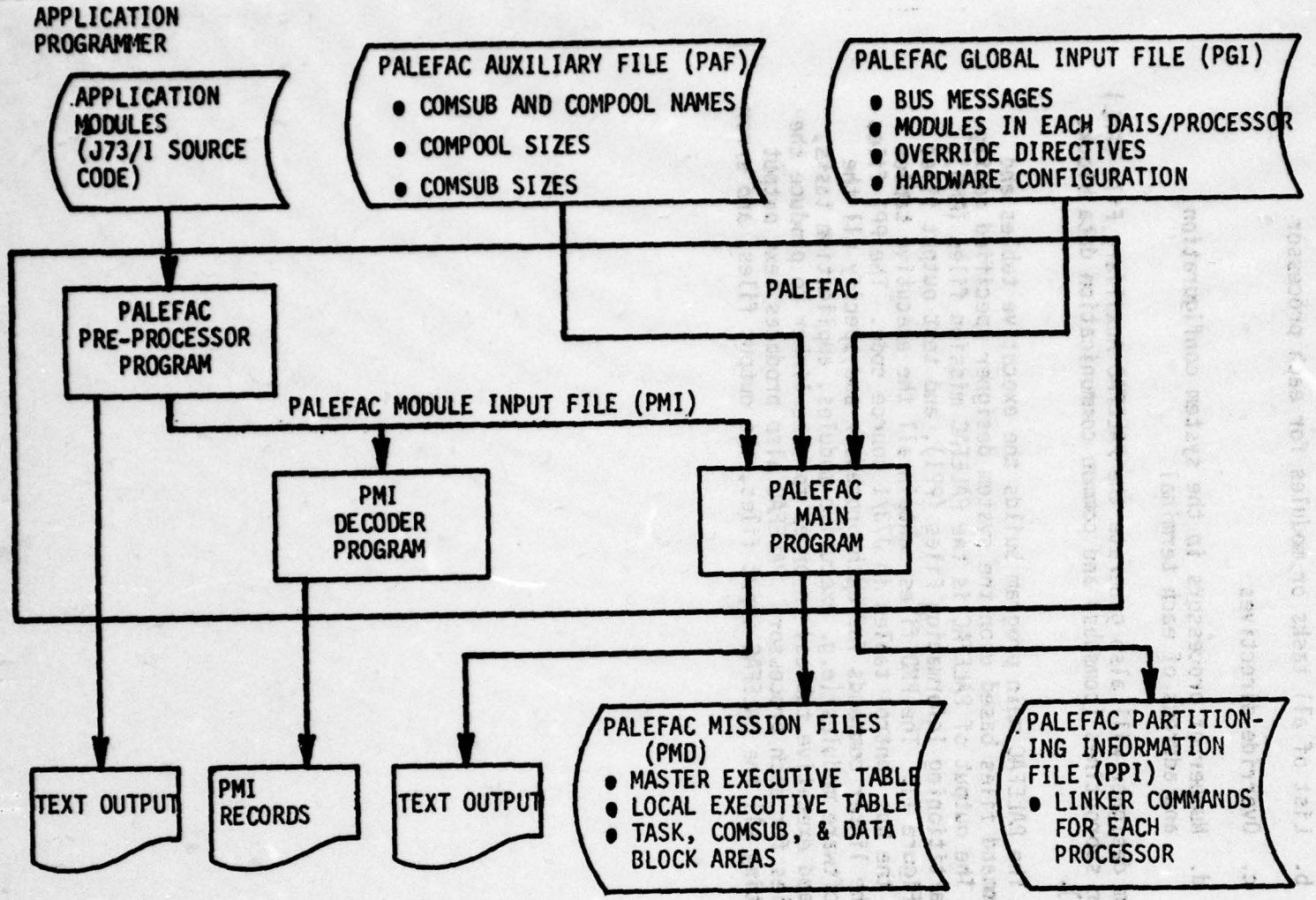


FIGURE 11. PALEFAC OVERVIEW

b. List of all tasks or modules for each processor

c. Override directives

d. Number of processors in the system configuration and address of each terminal

The system designer will also generate the PALEFAC Auxiliary File (PAL) for common subroutines (comsubs) and common communication data blocks (compools).

The PALEFAC main program builds the executive tables and linker command files based upon the system designer specified configuration. The output of PALEFAC is the PALEFAC mission files (PMD), PALEFAC Partitioning Information Files (PPI), and test output files shown in Figure 11. The PMD files contain all the executive tables including the bus control tables in J73/I source code. The PPI files contain the linker commands for each processor and specify all the mission software modules (e.g. executive modules, application tasks, comsubs, and executive tables) in order for the linker to produce the load modules for each processor. PALEFAC also produces text output files listing all the PALEFAC input files, the output files, and error messages.

4.0 DAIS SYSTEM CHARACTERISTICS AND FEATURES

The performance characteristics of DAIS include both system functional control characteristics as well as system architecture features and attributes included in the design of DAIS and its core elements. *These sections address the system control features and the primary characteristics which are unique and important to the DAIS concept.*

4.1 System Control Procedures

This section describes the system control procedures for DAIS. The DAIS architecture and integrated system design results in a special set of functional requirements, designated system control procedures, which include system startup/restart, control of the bus communication and utilization of redundancy in the event of core element failures. The system and bus control functions are distinguished from mission avionic functions which support the specific mission tasks. The mission avionic functions could be comprised of navigation, guidance, weapon delivery, communications, vehicle defense, target acquisition and track, auto pilot, stores management, and subsystem management.

A third important group, on-board test functions, utilize both hardware and software to isolate in-flight failures to allow utilization of redundancy, maintain a complete updated status of every equipment on the system, and provide isolation of failure to LRU levels with a minimum of auxiliary group equipment.

4.1.1 System Startup/Restart Operations

The overall objective of the system startup/restart procedures is to give the pilot control of the system configuration and to provide a verification of the hardware and software prior to entering normal system operations.

The pilot controls the system via the Processor Control Panel (PCP). The power enable switches allow him to remove a processor from the system by powering it down. The GROUND position of the startup switch permits him to specify an initial startup as opposed to a restart, indicated by the INFLIGHT position. The START button permits him to request the use of software already loaded, while the LOAD button allows him to force the reload of all software. Of course, the LOAD function is supported only when a system mass memory is configured in the system.

In addition to the pilot initiated restarts, the system will, under certain limited circumstances, perform an automatic restart. It is essential that an automatic restart have minimal impact to continue system operation. This consideration defines the need for a "warm start" initialization of software as opposed to the "cold start" associated with normal system startup.

"Cold start" assumes a fully operational configuration. It involves the complete initialization of executive and application software, and the assignment of initial values to events and data.

"Warm start" continues the current operational status. A partial initialization of the executive is performed, current data is maintained, and application tasks and events are assigned to a known, predetermined state.

The various startup/restart cases are summarized in Table 3.

As is evident from this table, it is important the pilot set the startup switch before powering up the processors. Otherwise, the power up would look like a transient recovery. Conversely, the pilot must return the switch to the INFLIGHT position at the completion of a normal startup. Failure to do so would allow a transient to bring the system to a stop.

4.1.1.1 Normal System Startup

Each processor is equipped with a read only memory (ROM)*, which is enabled when the power up condition is detected. The ROM contains sufficient instructions for the processors to identify the configuration, verify the software load modules and, if necessary, obtain the system loader from the system mass memory. The ROM may contain additional functions as desired for any specific implementation, but normally, considerations of stability and size would indicate only essentials should be in the ROM. The minimum ROM contents are dictated by the requirement that the ROM be capable of starting or restarting a system without a system mass memory.

As each processor senses power up, it enables the ROM and begins executing at location zero. Memory protection is established, interrupts are cleared, and self tests are performed on the processor and BCI.

At the completion of self tests, the second phase of the startup process is initiated. This is a coordinated procedure for establishing one processor as the controller of the startup and for identifying the configuration to be run. A processor is included in the configuration if it passed self test and it successfully communicates on the multiplex bus in response to commands from the controller. Also, during this phase of startup each processor determines the status of the software load in it's memory.

The third phase of the startup process, software verification/loading, is then entered. Based on the configuration which is operational, the controller accesses a startup configuration table to determine the load module required in each processor. If the required

*The ROM, in effect, overlays the first two thousand words of memory while envoked. This is a feature of the AN/AYK-15A processor and, since it is the preferred implementation, the startup procedures are described in this context. With the AN/AYK-15, or other processor not equipped with a ROM, the startup software must be permanently preserved in low memory and memory protected except as required when modifying interrupt vectors.

TABLE 3. START/RESTART CASES/OPTIONS

	PCP SWITCH SET TO GROUND	PCP SWITCH SET TO INFLIGHT
PCP START BUTTON DEPRESSED	NORMAL STARTUP • LOAD ONLY IF REQUIRED • COLD INITIALIZATION	SYSTEM RESTART (PILOT) • VERIFY RESIDENT SOFTWARE • NO RELOAD • WARM START
PCP LOAD BUTTON DEPRESSED	NORMAL STARTUP • FORCE RELOAD • COLD INITIALIZATION	SYSTEM RELOAD • FORCED RELOAD • COLD INITIALIZATION
POWER TRANSIENT	WAIT FOR START/LOAD	SYSTEM RESTART (TRANSIENT) • VERIFY RESIDENT SOFTWARE • NO RELOAD • WARM START

software is not present, it is loaded from mass memory, and the verification process repeated. When it is confirmed that all processors are properly loaded, BCIU addresses are reset to the logical ones required in the configuration. The Master processor assumes control and directs the set up for, and execution of, system initialization prior to starting system operation.

4.1.1.2 System Warm Start

System Warm Start is designed primarily as a response to a system wide power transient. The objective of a warm start is to get back on the air as quickly as possible. The procedure warm starts what it finds and verifies in the processors' memory and does not perform reloading.

As power is restored to each processor, it will enter the ROM, perform self tests, verify software and build a startup status block as in normal startup. The absence of any bus activity permits one of the processors to assume the role of control processor to manage the warm start. The fact that it is a warm start is determined by the INFLIGHT position of the start switch and the absence of both the discretes for the start button and the load button.

The startup configuration table defines the acceptable configurations for a warm start in their order of preference. If the procedure cannot find the configuration matching the initial load word (which by definition is the most preferred configuration for a warm start) it then goes through the alternate list trying to find a configuration in which all the load module ID's can be found in some processor.

The system implementer defines the preference order for a warm start by the order of the entries in the startup configuration table. One option, of course, is to make no entries of alternates, which has the effect of inhibiting warm starts for anything other than the initial load configuration. Care should be exercised to insure the recovery mode of a larger configuration is always included in the alternate list. Otherwise a transient would terminate a recovery mode of operation.

4.1.2 Normal System Operation

Normal system operation functions define the management of the processor/multiplex system configuration (Figure 2). One of the processors, designated the master processor, contains the master executive and controls the bus protocol via the BCIU operating with it. The application software is distributed among the remainder of the processors with each designated as a remote processor, and each containing a local executive. If required, another processor can be designated the monitor processor and would contain the backup master executive and application modules.

4.1.2.1 Bus Control Operation

The time division multiplexing (TDM) system provides for transmission of information from several terminals through a single communication system, with different terminals' information transmission staggered in time in accordance with MIL-STD-1553A.

The message structure, data format and command-response protocol required for the bus operation are defined in MIL-STD-1553A and are summarized in Figure 4. One exception was made to the protocol in the standard in order to facilitate error management and control. This pertains to terminal (or remote BCIU) response to receive data errors, and response to receive or transmit commands containing errors. For any detected data error, (i.e. incorrect word count, no data received, parity error, or improper Manchester coding) the terminal will not respond with a status word. The terminal will also ignore a command word not received entirely valid. These operations are required to avoid simultaneous RT response and to maintain error control within the master controller.

Message protocol is defined as that sequence of bus messages required to perform the normal multiplex synchronous and asynchronous operations and to support the error detection and recovery process. In order to insure minimum processor executive and multiplex bus overhead required to implement the message protocol, two concepts are introduced: mode commands and system control procedures (Figure 12).

System control procedures define the message protocol for the multiplexed data bus system. They serve to provide the executive software designer with the logical decisions required to manage normal system operation. They also define the required system flow to detect and recover from random bus message errors and selective terminal failure modes. The system control procedures serve to link the executive mission software, bus message protocol, and the terminal-to-subsystem interface management.

A time slice approach where messages occur on a minor cycle/major cycle basis is used to control bus message operations. Messages that are transmitted at specific iteration rates within a major cycle are defined as synchronous messages. Each synchronous message is assigned a period and phase for transmission. The number of minor cycles per major cycle is specified by the system designer, for example 128 minor cycles every second or major cycle. Messages that are to be transmitted on a demand or request basis and not at a specific phase and period are designated as asynchronous messages. Table 4 summarizes the bus control operations.

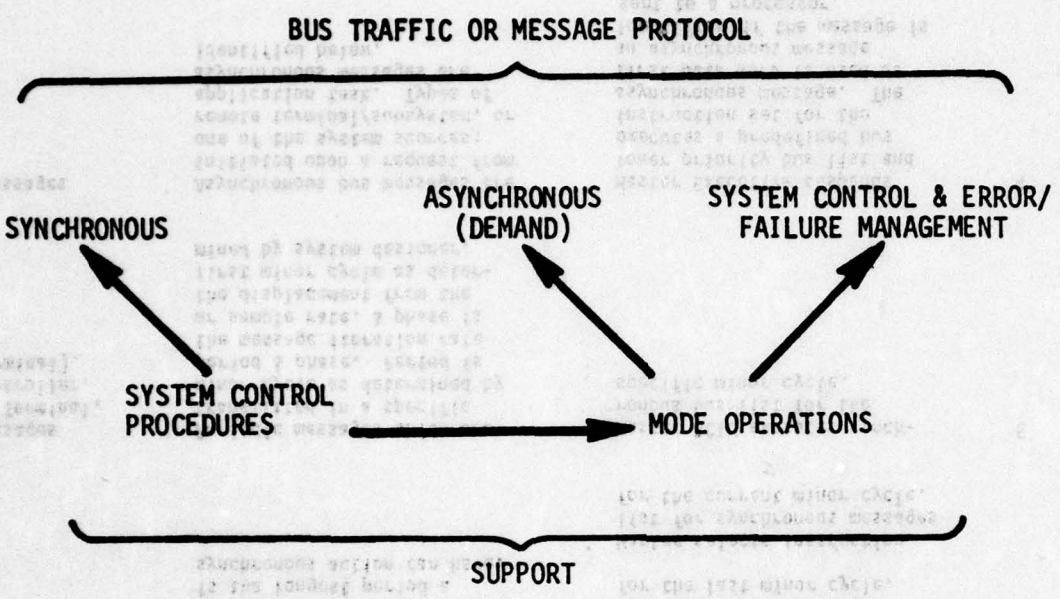


FIGURE 12. BUS OPERATIONS

TABLE 4. BUS CONTROL OPERATIONS
SHEET 1 OF 4.

BUS CONTROL OPERATIONS	FUNCTION	MECHANISM	PRIORITY	APPLICABLE RETRY PROCEDURES (SEE 4.3.3)
1. Minor Cycle Synchronous Message (Mode Commands)	• Signals Remote Processors to start a new minor cycle. • Minor Cycle is the shortest cyclic period, & major frame is the longest period a synchronous action can have.	• Master Executive initiates minor cycle sync messages upon expiration of minor cycle times & completion of synchronous bus messages for the last minor cycle. • Master selects instruction list for synchronous messages for the current minor cycle.	2	Class II Retry
2. Synchronous Messages (Controller to Terminal, Terminal to Controller, Terminal to Terminal).	• Periodic messages which are transmitted in a specific minor cycle as determined by period & phase. Period is the message iteration rate or sample rate, & phase is the displacement from the first minor cycle as determined by system designer.	• Master BCIU executes synchronous bus list for the specific minor cycle.	5	Class I Retry Class II Retry Class III Retry
3. Asynchronous Messages (General)	• Asynchronous bus messages are initiated upon a request from one of the system sources: remote terminal/subsystem, or application task. Types of asynchronous messages are identified below.	• Master Executive suspends lower priority bus list and executes a predefined bus instruction set for the asynchronous message. The first data word is used as an asynchronous message identifier if the message is sent to a processor.	4	Class I Retry Class IV Retry Class V Retry Class VI Retry

TABLE 4. BUS CONTROL OPERATIONS
SHEET 2 OF 4

BUS CONTROL OPERATIONS	FUNCTION	MECHANISM	PRIORITY	APPLICABLE RETRY PROCEDURES (SEE 4.3.3)
3A. Interprocessor Async Message (Controller to Terminal, Terminal to Controller, Terminal to Terminal).	Transfer asynchronous data or executive service requests between processors.	Local executive (if not in master processor) signals Master Executive by placing a nine bit Asynchronous request Vector (ARV) in Remote BCIUs status code register. Master receives ARV in next Status Word Response from that BCIU. (If the local executive is located in the master processor, the ARV is passed directly.) The ARV points to a predefined bus instruction set which is executed to effect the message transmission.	4	Class IV Retry Class V Retry Class VI Retry
3B. Critically Timed Messages (Controller to Terminal).	Application tasks in any processor request a message be transmitted to a specific remote terminal at a specified future time.	When critical timer expires, Master suspends lower priority bus list and sends predefined message. If application task is in remote processor, a prior interprocessor asynchronous message to the master is required to identify the critically timed message. and time it is to be sent.	3	Class I Retry Class II Retry Class III Retry

TABLE 4. BUS CONTROL OPERATIONS
SHEET 3 OF 4

BUS CONTROL OPERATIONS	FUNCTION	MECHANISM	PRIORITY	APPLICABLE RETRY PROCEDURES (SEE 4.3.3)
3C. Remote Terminal Requested Asynchronous Message (Terminal to Controller, Terminal to Terminal)	• Asynchronous request from Remote Terminal is initiated by subsystem for transmission of data to a processor.	• Subsystem flags request to RT which sets activity bit in status word and activity register. Master sends Interrogate Activity Register Mode Code and determines pending request. Master executes bus instructions to transmit the data. Master resets the RT subsystem lockout with a Reset Serial Input Channel Mode Code. The first data word transmitted is the RT Last Command Register which identifies the RT asynchronous message.	4	Class I Retry Class IV Retry
3D. Processor to Remote Terminal Message (Controller to Terminal, Terminal to Terminal)	• Application task in processor request a message be transmitted to a specific remote terminal.	• Master executes predefined bus instruction set. If application task is in a remote processor, local executive signals Master Executive with ARV via status word to identify instruction set.	4	Class I Retry Class II Retry Class III Retry Class V Retry Class VI Retry

TABLE 4. BUS CONTROL OPERATIONS
SHEET 4 OF 4

BUS CONTROL OPERATIONS	FUNCTION	MECHANISM	PRIORITY	APPLICABLE RETRY PROCEDURES (SEE 4.1.4)
4. Message Error and Terminal Failure Analysis Operation (Mode Command)	• Transmission of various mode commands to support message error analysis and retry, and terminal failure analysis.	• Master executes assorted Mode Code Operations to confirm the error and determine status of receiver and/or transmitter.	1	Class I Retry
5. Status Polling Messages (Mode Command Messages)	Periodically poll each terminal for status and request for asynchronous messages.	• When Master Executive has no other messages to be transmitted, a predefined bus list of Transmit Status Mode Command will be initiated. Each terminal will be polled at least once each major frame or at a periodic rate as required to service terminal asynchronous requests.	6	Class I Retry
6. System Mass Memory Operations	• Application Task in any processor requests data be transferred to/from system mass memory.	• When SIL and other asynchronous messages are complete, Master commands data transfer. If Application task is in remote, prior transfer of data with request is required for WRITE.	6	Special Retry

Bus control operations provide for major/minor cycle synchronization by transmission of minor cycle sync mode command messages to each remote BCIU/processor, and for synchronous and asynchronous bus message transmissions as shown in Figure 13. The synchronous bus messages are controlled by a predefined bus instruction list. The bus instruction list is part of the Master Executive preloaded tables and contains an instruction pair for each synchronous message that is transmitted on the data bus. Since each message is not transmitted every minor cycle, the instruction lists are organized so that the proper instruction pairs can be linked together at the start of each minor cycle to form the complete instruction list for that minor cycles' period and phase. The link instruction pairs are dynamically set by the master executive to point to the proper block of instructions to be performed during the current minor cycle.

Each data word on the bus is transferred from the bus controller (whether master or remote) to the processor's memory in real time via a direct memory access channel. The address at which each message block begins is constructed by fetching first the receive (or transmit) pointer word and by combining the current base address, the transmit/receive bit, and the subaddress into the pointer address. Consecutive data words are stored in contiguous memory locations until the word count for that particular message is exhausted. For all receive operations, each message block is preceded by a generated tag word which contains the current minor cycle number, word count of the received message, and any message error relative to the received message.

Asynchronous messages are scheduled by the master executive based upon requests from remote terminals, or application tasks in either the master or remote processors or from the master executive to perform message error or terminal failure operations. Subaddress 31 is dedicated to asynchronous processor/BCIU messages. After receiving or transmitting an asynchronous message, the BCIU interrupts the processor and the local executive updates the pointer word for storing and fetching the data for the asynchronous message. The first word of all asynchronous messages is used to designate the source of the message to the receiving processor. This asynchronous message identifier is used to direct the local executive to a predefined table containing information to process the data or perform the executive services action required.

There are several types of asynchronous bus message operations based upon the resource making the request and protocol to process the request as defined in Table 4. Upon receiving the request, the master executive will cause the BCIU to halt after completing the current message transmission, and link in a predefined or generated BCIU instruction pair to initiate the asynchronous message transmission.

---

*Refers to either the processor & BCIU, or integrated processor/BCIU.

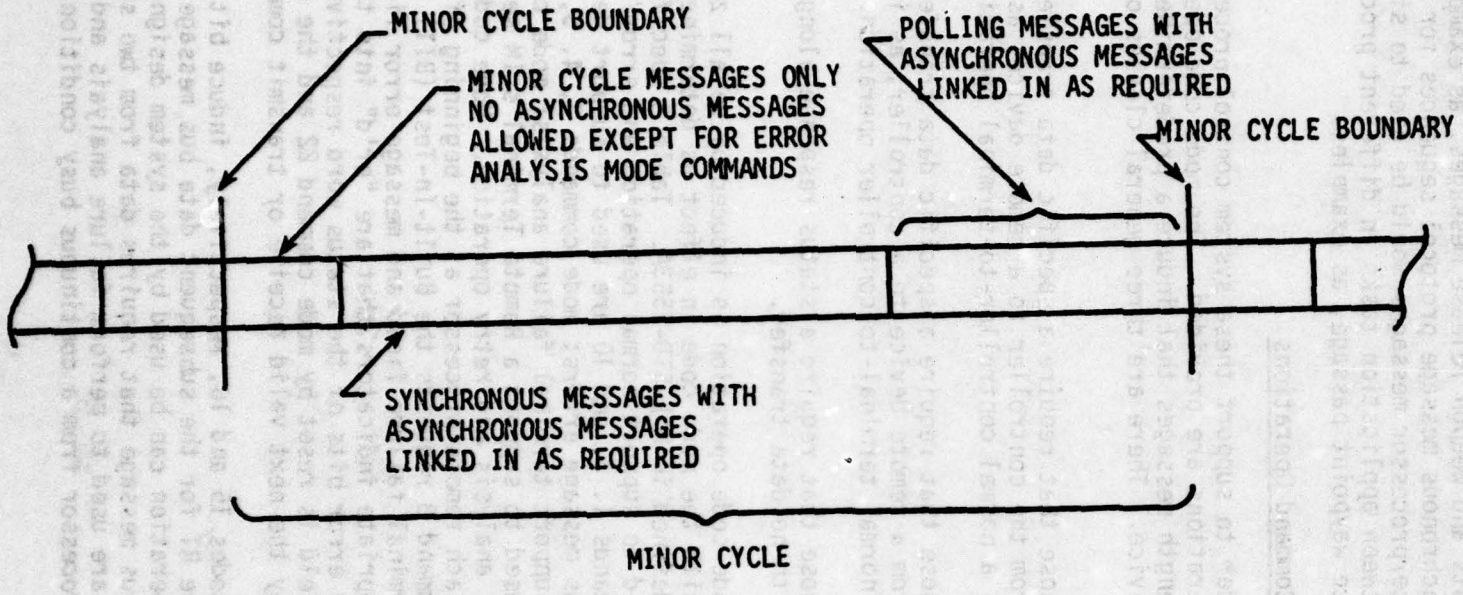


FIGURE 13. MESSAGE TRANSMISSION DURING A MINOR CYCLE

Figure 14 shows the message protocol sequence for a subsystem connected to an RT requesting service. Asynchronous operations would be used for pilot panel inputs and weapon release messages, as examples. Figures 15 and 16 show asynchronous message protocol sequences for interprocessor operations. Interprocessor messages would be used to signal master mode changes between application tasks in different processors, and steering/guidance waypoint passage, as examples.

4.1.2.2 Mode Command Operations

In order to support these system control procedures, a set of mode code operations are provided. The mode code operations are defined fixed-length messages that induce a predetermined response from a remote device. There are three general classes of mode operations:

a. Those that require a specific data transfer from the controller to a remote device, as in a normal controller-to-terminal operation.

b. Those that require a specific data transfer from a remote device to a controller, as in a normal terminal-to-controller operation.

c. Those that require a status response along with no data transfer.

The mode code operation is induced by an all zero subaddress command field; with the mode code in effect as determined in the word count field as defined in MIL-STD-1553A. Table 5 specifies the mode commands required to support normal operation and error/failure management. Mode commands 1, 3, and 10 are used to support detection and recovery from bus message errors; mode commands 3, 4, 5, 6, 7, 9, and 22 are used to support terminal failure analysis; mode commands 11, 12, 13, and 14 are used to support a Remote Terminal's IM serial/digital channel request, analysis and retry operation. Mode code 17 is used to synchronize each remote processor at the beginning of a minor cycle. Mode command 3 requests the Built-In-Test (BIT) register consisting of a terminal failure field and message error field, each containing appropriate indicators that are "or'd" into the terminal fail and message error bits of the status word respectively. The terminal fail field is reset by mode command 22 and the message error field is reset by the next valid receive or transmit command.

Mode codes 15 and 16, respectively, induce bit and word mask operations in the RT for the subsequent data bus message. The bit mask and word mask operation can be used by the system designer when partitioning a data bus message that requires data from two sources. Mode codes 20 and 21 are used to perform failure analyais and recovery of a remote BCIU/Processor from a continuous busy condition.

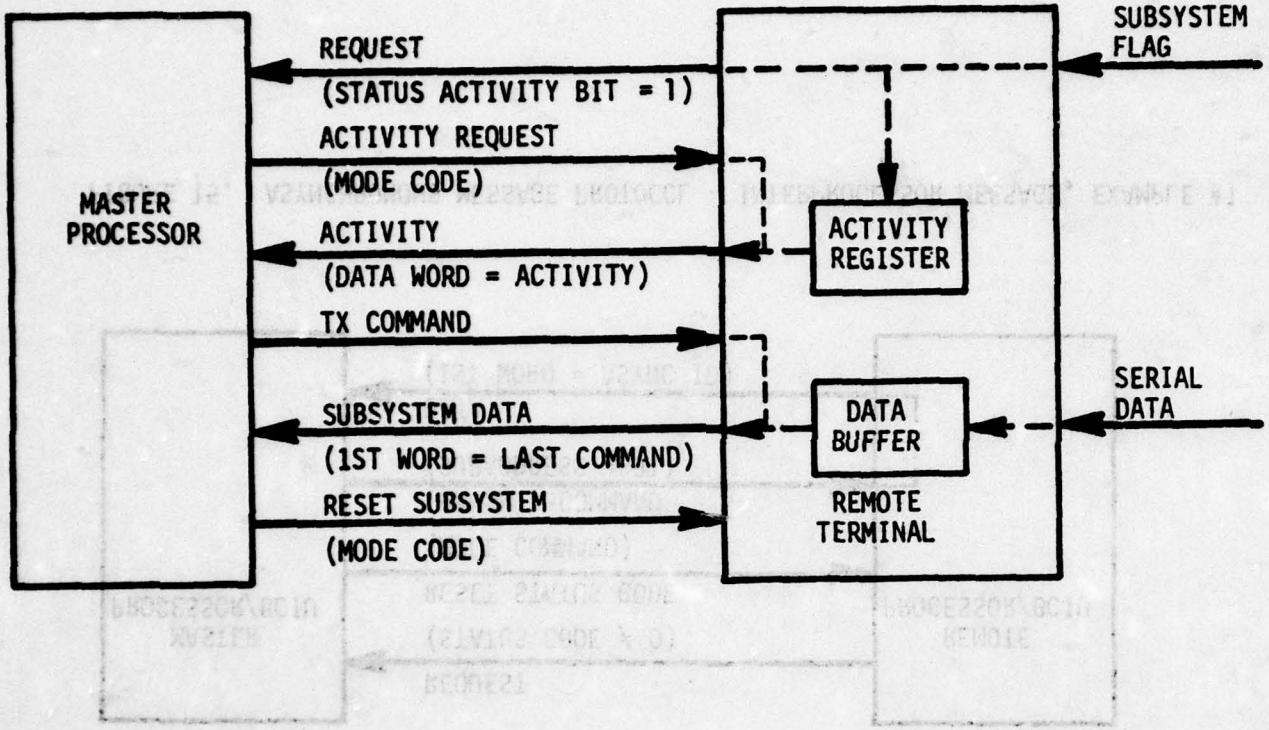


FIGURE 14. ASYNCHRONOUS MESSAGE PROTOCOL - REMOTE TERMINAL REQUEST

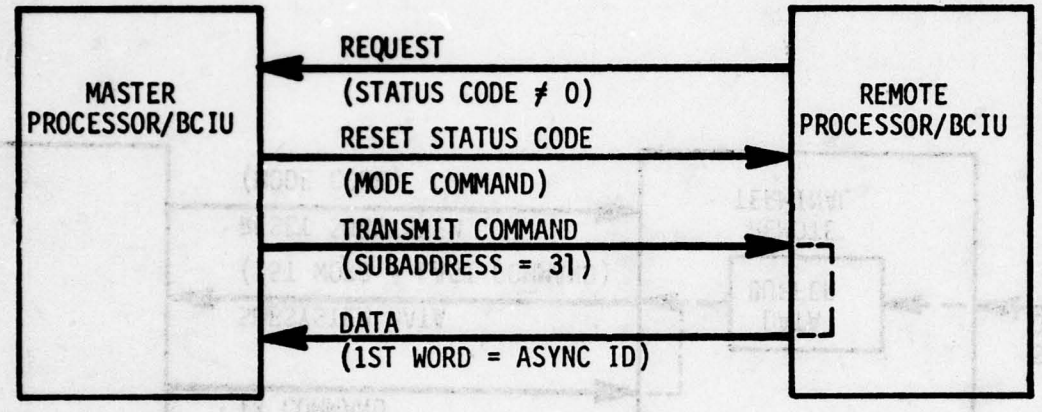


FIGURE 15. ASYNCHRONOUS MESSAGE PROTOCOL - INTERPROCESSOR MESSAGE, EXAMPLE #1

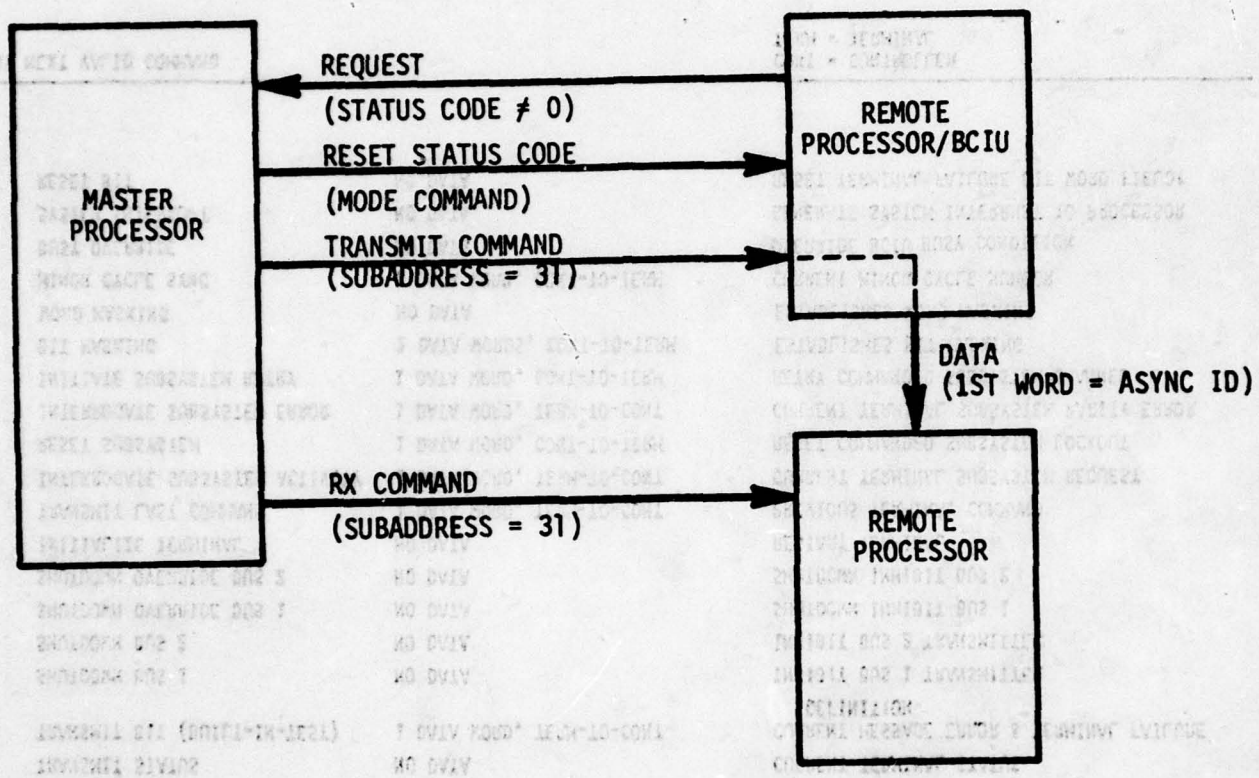


FIGURE 16. ASYNCHRONOUS MESSAGE PROTOCOL - INTERPROCESSOR MESSAGE, EXAMPLE #2.

TABLE 5. MODE CODE COMMANDS

MODE CODE NUMBER	DESCRIPTION	PROTOCOL	OPERATION	VALID FOR:
1	TRANSMIT STATUS	NO DATA	CURRENT TERMINAL STATUS	BCIU/RT
3	TRANSMIT BIT (BUILT-IN-TEST)	1 DATA WORD, TERM-TO-CONT	CURRENT MESSAGE ERROR & TERMINAL FAILURE DEFINITION	BCIU/RT
4	SHUTDOWN BUS 1	NO DATA	INHIBIT BUS 1 TRANSMITTER	BCIU/RT
5	SHUTDOWN BUS 2	NO DATA	INHIBIT BUS 2 TRANSMITTER	BCIU/RT
6	SHUTDOWN OVERRIDE BUS 1	NO DATA	SHUTDOWN INHIBIT BUS 1	BCIU/RT
7	SHUTDOWN OVERRIDE BUS 2	NO DATA	SHUTDOWN INHIBIT BUS 2	BCIU/RT
9	INITIALIZE TERMINAL	NO DATA	RESTART TERMINAL	BCIU/RT
10	TRANSMIT LAST COMMAND	1 DATA WORD, TERM-TO-CONT	PREVIOUS TERMINAL COMMAND	BCIU/RT
11	INTERROGATE SUBSYSTEM ACTIVITY	1 DATA WORD, TERM-TO-CONT	CURRENT TERMINAL SUBSYSTEM REQUEST	RT
12	RESET SUBSYSTEM	1 DATA WORD, CONT-TO-TERM	RESET COMMANDED SUBSYSTEM LOCKOUT	RT
13	INTERROGATE SUBSYSTEM ERROR	1 DATA WORD, TERM-TO-CONT	CURRENT TERMINAL SUBSYSTEM PARITY ERROR	RT
14	INITIATE SUBSYSTEM RETRY	1 DATA WORD, CONT-TO-TERM	RETRY COMMANDED SUBSYSTEM CHANNEL	RT
15	BIT MASKING	2 DATA WORDS, CONT-TO-TERM	ESTABLISHES BIT MASKING	RT
16	WORD MASKING	NO DATA	ESTABLISHES WORD MASKING	RT
17	MINOR CYCLE SYNC	1 DATA WORD, CONT-TO-TERM	CURRENT MINOR CYCLE NUMBER	BCIU
20	BUSY OVERRIDE	NO DATA	OVERRIDE BCIU BUSY CONDITION	BCIU
21	SYSTEM INTERRUPT	NO DATA	GENERATE SYSTEM INTERRUPT TO PROCESSOR	BCIU
22	RESET BIT	NO DATA	RESET TERMINAL FAILURE BIT WORD FIELD*	BCIU/RT

*ME FIELD RESET BY NEXT VALID COMMAND

CONT = CONTROLLER
TERM = TERMINAL

Table 5 indicates which mode code operations are required in a remote device in order to be compatible with the DAIS system control procedures and master executive. Utilization of the other mode code operations is optional as determined by the system designer for the specific system application.

Table 6 shows the use of the mode commands in relationship to the system control functions.

4.1.3 Application Software Executive Services

The DAIS Local Executive provides services for the application software to control the operation of the application functions and data in each individual DAIS processor. The application software elements which are recognized by the local executive software are Tasks, Comsubs, Compool Blocks, and Events, defined as follows:

a. Tasks are programs or processing modules which can either be controllers or calculators. Application software is organized in tasks (program modules) with each performing a function as required for the specific system application. Each task contains executable code and local data.

b. Comsubs are common subroutines which differ from tasks in that they are required to be re-entrant. Comsubs can be used by many tasks, so that one task using a particular comsub can be suspended in the middle of the comsub routine by another task using the same comsub.

c. Compool Blocks provide data communication between tasks and the external world or equipment. A buffer system is provided by the use of "global copies" and "local copies" to prevent a compool block from being read when it has been partially updated. Every task, except "privileged" tasks, interface with a local copy while performing its calculations. The local copy is updated from the global copy with the read statement and the local copy updates the global copy by the write statement. A special statement (trigger) is used for a critically timed compool block.

In a "privileged" task, communication is allowed directly with the global copy in the processor in which the task resides since the privileged task cannot be interrupted by another task or by the Executive.

Since the read, write, and trigger statements invoke Executive services that operate in the Privileged Mode, this guarantees a task will not reference partially updated data.

TABLE 6. MODE CODE RELATIONSHIP TO SYSTEM FUNCTIONS

DAIS SYSTEM FUNCTION / SUPPORTING MODE CODES	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23-31
		NOT USED	TRANSMIT STATUS	RESET STATUS CODE	TRANSMIT BIT WORD	SHUTDOWN BUS 1	SHUTDOWN BUS 2	SHUTDOWN OVERRIDE BUS 1	SHUTDOWN OVERRIDE BUS 2	INITIATE SELF TESTS	INITIALIZE TERMINAL	TRANSMIT LAST COMMAND	INTERROGATE ACTIVITY REGISTER	RESET SERIAL INPUT CHANNEL	INTERROGATE MODULE ERROR REGISTER	INITIATE SERIAL CHANNEL I/O	BIT MASKING	WORD MASKING	NO OPERATION	MASTER FUNCTION INTERRUPT	UNDEFINED	BUSY OVERRIDE	SYSTEM INTERRUPT	RESET BIT WORD	UNDEFINED
NORMAL SYSTEM INITIALIZATION		X	X							X													X	
MINOR CYCLE SYNCHRONIZATION																			X					
SYNCHRONOUS MESSAGE OPERATION																X	X							
ASYNCHRONOUS MESSAGE OPERATION		X	X									X	X		X	X	X							
ERROR/RETRY MANAGEMENT		X		X						X	X			X	X		X	X			X	X	X	
POWER CONTROL					X	X	X	X															X	

The application software requests service of the executive through real time statements. Real time statements are used by tasks to control the state of other tasks, control values of events, and control reference to compool blocks. These are J73/I defined statements which the application programmer will use during the development of mission software as follows:

Schedule Statement
Cancel Statement
Terminate Statement
Wait Statement
Signal Statement
Read Statement
Write Statement
Trigger Statement
Broadcast Statement

The first four statements shall be used to control task states. At any one time, a task shall be in one of these states and the method of transitioning from one state to another state is shown in Figure 17.

a. Schedule Statement. Schedule statements are used to place an uninvoked task into an invoked state. An invoked task becomes active and dispatchable when the required event conditions are satisfied as shown in Figure 17. An invoked task is placed in execution by the Executive if it has the highest priority among the tasks which are dispatchable.

A schedule statement includes the name of the task to be scheduled, the mode of task (privileged or normal), priority of the task, and event conditions to be satisfied.

b. Cancel Statement. The Cancel Statement is used by one task to place another task in the Uninvoked State. If a task is cancelled, all the lower branches or sons of the tasks in the control tree structure are cancelled.

c. Terminate Statement. The Terminate Statement is used by one task to place another task in the inactive, but invoked state. If a task is terminated, all the lower branches or sons of the tasks in the control tree structure are also terminated.

d. Wait Statement. Wait Statements are used by tasks to place themselves into the Wait State pending occurrance of one of the following conditions:

1. Absolute Time
2. Relative Time
3. Latched Event
4. Unlatched Event

For the latched or unlatched events, the task remains in the Wait State until the specified event, either on or off, occurs.

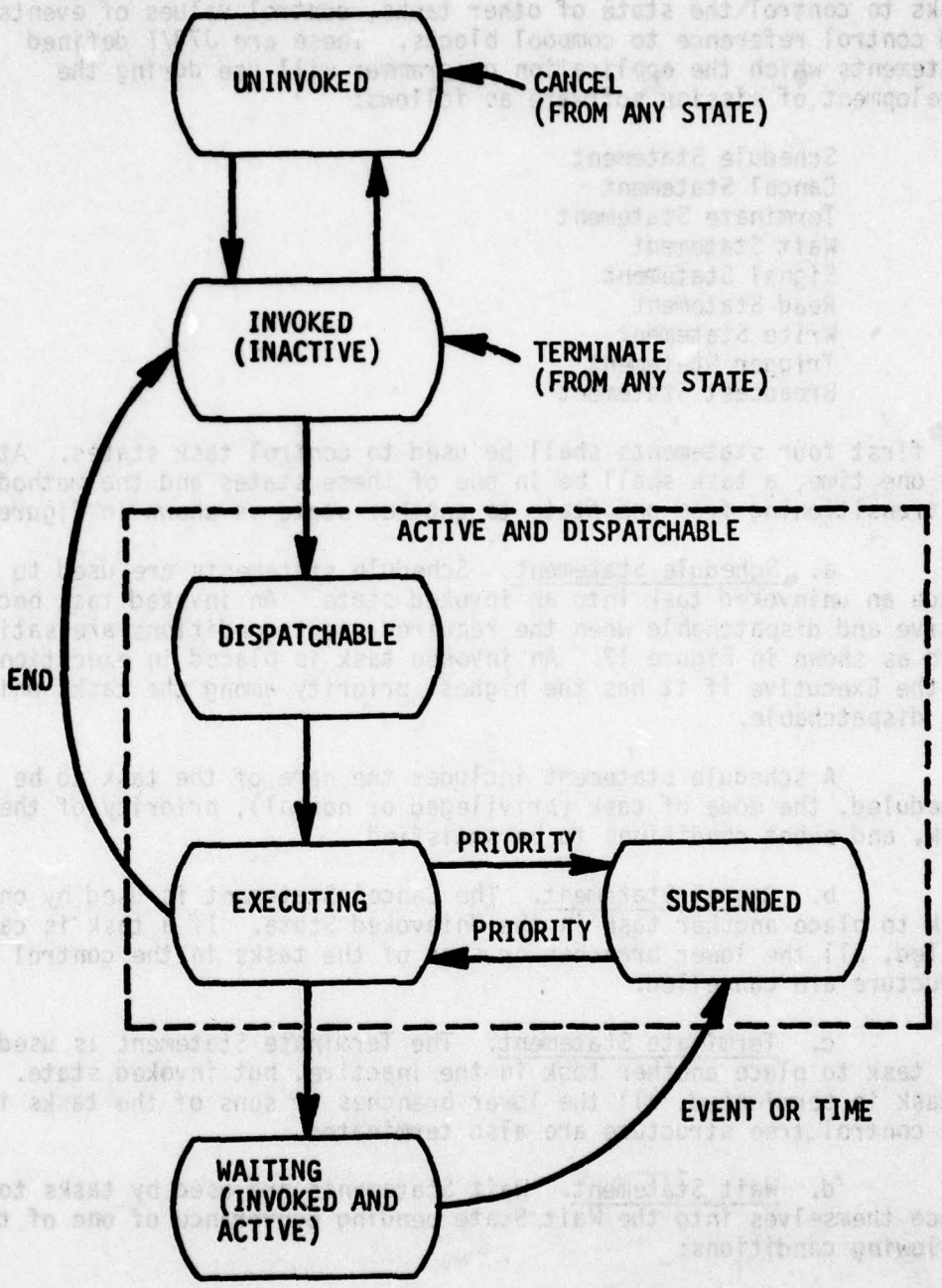


FIGURE 17. TASK STATE DIAGRAM

e. Signal Statement. Signal Statements are used by tasks to set a specified event to a specified value, either on or off.

f. Read Statement. Read Statements are used by tasks to copy the values of specified compool blocks (global copies) into a local copy to be used by the task.

g. Write Statement. Write Statements are used by the tasks to copy the values of specified local copy into the specified compool block (global copy). The Write Statement causes global copies in other processors or virtual copies in remote terminals to be updated, and causes signalling of an event associated with the compool block.

h. Trigger Statement. The Trigger Statements are used by a task to send a specified critically timed compool block (data) to a specified remote terminal at a specified time. This causes a critically timed asynchronous bus message operation to be performed where the local copy of a specified compool block is sent to a global copy in the Master Processor; and where, at the specified time, the data is transmitted asynchronously to the specified remote terminal.

i. Broadcast Statement. The Broadcast Statement is used only by a privileged mode task. After updating a global copy of a compool block, the broadcast statement is used to cause the asynchronous update of remote and/or virtual copies of the compool block.

4.1.4 Error and Failure Management.

Bus message error management and terminal failure management are performed by the master BCIU and the master executive. In addition, each terminal on the multiplex data bus provides mechanisms for detecting and recording message error conditions and terminal failures. Message error conditions and terminal failure conditions are recorded in the Built-In-Test (BIT) word which can be obtained by the master with a mode command for message error and terminal failure analysis. Also, each terminal records in a register (Last Command) the last valid command received by that terminal which also can be obtained by the master with a mode command for message error analysis.

The master executive and BCIU centrally manages the message protocol to recover from message errors and reconfigure for terminal failures as follows:

a. Monitors the message sequence to determine successful/unsuccessful completion.

b. Analyze error response (data error or invalid status, or no response to command) and determine type of retry procedure for the unsuccessful message.

c. Perform retry of unsuccessful message, first on the same data bus path.

d. Analyze error response if not successful, and retry the unsuccessful message on redundant data bus path, if available.

e. Perform failure analysis if retry of message on alternate bus is unsuccessful and determine recovery or backup approach.

f. Update the system configuration tables based upon the error conditions and failures.

Table 7 describes the operation of the master BCIU to various types of bus message errors relative to the three classes of data transfer protocol. An invalid word (command, data or status) would include one or more of the following conditions:

- Parity error
- Incorrect or missing sync
- Incorrect bit count
- Invalid manchester format

A remote receiver will not respond with a status word after detecting a message error. Both remote receivers and transmitters ignore invalid commands. The last command (i.e. the last valid received command) register and the message error field of the Built-In-Test register are stored as shown in Table 7. As message data is received by a BCIU, it is stored in the processor memory via DMA. If received unsuccessfully, the message error bit in the tag word is set. As message data is received by an RT, the data is temporarily stored. If received successfully, the data is transferred to the respective subsystems; otherwise, it is not transferred to the subsystems.

All bus errors are grouped into a set of message error indications by the master BCIU as shown in Table 7. These error indications will generate an interrupt to the master processor which contains the service routine required to initiate a retry procedure shown in Table 8. Depending upon the type of bus message operation and master BCIU message error indication, the appropriate retry procedure as shown in Table 9 would be initiated.

Error analysis and message retry procedures are performed by the Master Processor/BCIU. Error analysis is based on:

- The type of bus message operation
- The status word(s) received from the transmitting and/or receiving terminal involved in the operation, and

TABLE 7. MESSAGE ERROR RESPONSE

ERROR CONDITION	REMOTE RECEIVER		REMOTE TRANSMITTER		MASTER BCIU MESSAGE ERROR INDICATIONS	BCIU STORES DATA IN PROC MEMORY (ERROR TAGGED	REMOTE TERMINAL OUTPUT DATA TO SUBSYSTEM
	STATUS RESPONSE	INFORMATION RECORDED	STATUS RESPONSE	INFORMATION RECORDED			
Invalid Command Word							
Cont-to-Term	No	No	--	--	No Receiver Status Word	No	No
Term-to-Cont	--	--	No	No	No Transmit Status Word	---	--
Term-to-Term	No	No	No	No	No Receiver/Transmit Status Word	No	No
Invalid Data Word							
Cont-to-Term	No	Last Cmd Built-In-Test	--	--	No Receiver Status Word	Yes/Tagged	No
Term-to-Cont	--	--	Yes	Last Cmd	Invalid Data Word	Yes/Tagged	--
Term-to-Term	No	Last Cmd Built-In-Test	Yes	Last Cmd	No Receiver Status Word	Yes/Tagged	No
Word Count Error or No Data Received							
Cont-to-Term	No	Last Cmd Built-In-Test	--	--	No Receiver Status Word	Yes/Tagged	No
Term-to-Cont	--	--	Yes	Last Cmd	Word Count Error	Yes/Tagged	No
Term-to-Term	No	Last Cmd Built-In-Test	Yes	Last Cmd	No Receiver Status Word	Yes/Tagged	No
Invalid Status Word							
Cont-to-Term	Yes	Last Cmd (No Error)	--	--	Invalid Status	Yes	Yes
Term-to-Cont	---	--	Yes	Last Cmd (No Error)	Invalid Status	Yes/Tagged	---
Term-to-Term	Yes	Last Cmd (No Error)	Yes	Last Cmd (No Error)	Invalid Status	Yes/Tagged	Yes
Invalid Mode Command							
Cont-to-Term	Yes	Built-In-Test	--	--	Receiver Status Word with 'ME' Bit Set	---	---
Term-to-Cont		--	Yes	Built-In-Test	Transmit Status Word with 'ME' Bit Set	---	---

TABLE 8. MESSAGE RETRY PROCEDURES

CLASS RETRY	PURPOSE	PROCEDURE	APPLICABLE MESSAGE OPERATION
Class I Retry (Auto Retry)	Immediate retry of the bus message 1, 2, or 3 times.	BCIU automatically retries bus message on same bus 1, 2, or 3 times (as specified in bus instruction) before presenting error interrupt to processor.	Most Synchronous Operations, and most asynchronous operations not involving a remote processor.
Class II Retry (Careful Retry)	Retry of the bus message only if the terminal had not successfully received the message. (Terminal required to have Last Command capability.)	Master processor obtains the Last Command and BIT registers via Mode Commands. If the last command is not the same as the bus message, or an error is indicated, the bus message is retransmitted. Otherwise, the master continues with the next scheduled bus message.	Synchronous or asynchronous messages to the remote terminal/subsystem when repeated transmission of the same data will cause incorrect operation or degrade performance of the subsystem.
Class III Retry (Message Sequence Retry)	Retry of a bus message only after previous messages in the sequence have been repeated.	Master restarts the entire message sequence when error occurs with any message in the sequence.	Synchronous or asynchronous message to a subsystem when the entire message sequence must be received for correct operation of the subsystem.
Class IV Retry (Straight Retry)	Confirm the Remote did not receive the message properly and Retry message after BCIU has advanced Instruction Address Register.	Analyze Last Command, and BIT word as above, then reduce the value of the IAR by two and restart BCIU.	Asynchronous Messages from a transmitter (Master or RT) that does not need to be realigned.
Class V (Realign Transmitter)	Determine if the transmitter must be realigned prior to repeating the message.	Save the IAR minus 2. Obtain the transmitter's Last Command and status via mode code 10. If Last Command is correct, send a retransmit message to realign the Remote Transmitter. Restore the IAR and repeat the message.	Asynchronous Messages from Remote Processor to Master or an RT insensitive to repeated data.
Class VI (Confirm and Realign)	Confirm that message was not received properly, then determine if the transmitter must be realigned prior to repeating the message.	Save the IAR minus 2. Obtain the receiver's Last Command and BIT words via mode codes 10 and 3. If Last Command is correct and no message error indicated, continue with next bus message (i.e. do not retry). Otherwise, obtain the transmitter's last command and status via mode code 10. If Last Command is correct, send a retransmit message to realign the Remote Transmitter. Restore the IAR and repeat the message.	Asynchronous Messages, Remote to Remote, or Remote to RT when a careful retry is required by subsystem.

TABLE 9. RETRY PROCEDURE FOR EACH MESSAGE OPERATION

BUS MESSAGE OPERATIONS	CONTROLLER TO TERMINAL		TERMINAL TO CONTROLLER		TERMINAL TO TERMINAL		
	Receive Status Exception (RSEX) - Message Error Bit = 1*	Receive Status Error (RSE): • Not Received • Invalid Status • Parity Error • Incorrect Address	Data Word Errors: • No Data Received • Word Count High or Low (Incomplete Data) • Data Word Parity Error • Invalid Data	Transmit Status Error (XSE): • Not Received • Invalid Status • Parity Error • Incorrect Address	Receive Status Exception (RSEX) - Message Error Bit = 1*	Receive Status Error (RSE): • Not Received • Invalid Status • Parity Error • Incorrect Address	Transmit Status Error (XSE): • Not Received • Invalid Status • Parity Error • Incorrect Address
Minor Cycle Synchronization	Class II Retry	Class II Retry	---	---	---	---	---
Synchronous Messages	Class I, II, or III Retry	Class I, II, or III Retry**	Class I Retry	---	Class I, II, or III Retry**	Class I, II, or III Retry**	---
Asynchronous Messages							
Interprocessor Message (Processor to Processor)	Class IV Retry	Class IV Retry	Class V Retry	---	Class VI Retry	Class VI Retry	---
Critically Timed Message (Controller to Remote Terminal)	Class I, II, or III Retry**	Class I, II, or III Retry**	---	---	---	---	---
Remote Terminal Requested Message (From RT Only)	---	---	Class I Retry	---	Class IV Retry	Class IV Retry	---
Processor to Remote Terminal Message	Class I, II, or III Retry**	Class I, II, or III Retry**	---	---	Class V or VI Retry**	Class V or VI Retry**	---
Error or Terminal Failure Analysis Mode Commands	Class I Retry	Class I Retry	Class I Retry	---	---	---	---
Status Polling Mode Command Messages	---	---	---	Class I Retry	---	---	---

--- Not Applicable
*For most terminals this case does not exist, since they signal an error by not returning a status word.
**User's defined Retry.

- The internal status register (ISR) in the Master BCIU indicating if a status word was received from each terminal involved in the operation.

The method of retry is based on specifications of the system designer, the type of message operation, and the manner in which the error is detected. The six classes of retry are defined in Table 8.

A Class I retry is specified by including a non-zero value in the retry field of the bus instruction. This is the standard retry procedure for most messages. It is performed automatically by the Master BCIU without processor intervention.

A Class II or Class III retry is specified by including a NO OP instruction following the bus list instruction to send the message. Class II (careful retry) is utilized if a subsystem would be degraded or function improperly when a message is repeated. Class III (sequence retry) is utilized when a series of messages must be successfully transmitted for proper subsystem oepration, such as to arm the Station Logic Unit (SLU).

Classes IV, V, and VI are utilized with asynchronous messages, and depend on the terminals involved and whether or not they recognize that a message error occurred. As with Class II, the receiver could be harmed by a repeated message. In addition, the transmitter, if not advised of the error, would send its next asynchronous message when a repeat is attempted.

Class IV is invoked when the transmitter need not be realigned. Class V is used when it is uncertain if the transmitter saw the error and, finally, Class VI applies when it is uncertain if either the transmitter or receiver saw the error.

Terminal failures are also detected by self-tests performed within each BCIU and Remote Terminal by the micro-controller. Each BCIU and RT while not participating in data transfer on the bus, performs an internal background self-test. These include register data-pattern wraps and verification of basic micro-code operations. Results of these tests are encoded into the appropriate fields of the Built-In-Test Word. In addition, the master processor may elect to perform a comprehensive self-test included in the master mode BCIU. This contains all those tests performed in background as well as a full parity sum check of all micro instructions.

When a redundant element in a terminal (data bus interface 1 or 2) or a terminal (data bus interface 1 and 2) is declared as failed, the master executive, if possible, maintains the system in an operational state by utilizing the redundant data bus path, or perform backup or recovery operations by automatically switching to a redundant unit.

A terminal failure is indicated if a message is not successfully transmitted after retries have been attempted on both buses. Terminal Failure Analysis is invoked by the Configuration Management function when it detects an accumulation of errors in excess of a parameter defined by the system designer for the specific terminal.

The function of this procedure is to confirm that the terminal has failed or, if possible, to clear the error condition. In either case, the results of this analysis are reported to Configuration Management, which has the responsibility of determining if communications should be inhibited with the entire terminal, if additional retries should be attempted, or if selected subsystems should be declared as failed.

The analysis performed differs slightly for the different types of terminals. The DAIS built-in tests provide the primary mechanism for the failure analysis. In addition, for a BCIU there exists processor/BCIU interface tests which are exercised. The handling of the Master BCIU differs in that it may be accessed directly via PIO rather than using mode commands.

4.1.5 Configuration Management

The repeated failure of a bus message is reported to the Configuration Management function for tabulation and analysis. A message which succeeds on immediate retry is not reported. (In fact, the processor is not even aware of the error if a Class I retry succeeds.) Reported errors are tabulated against a terminal address with separate counters accumulating errors on the two redundant communications paths.

The overall interaction of the assorted system functions involved in declaring or responding to a device failure is shown in simplified form in Figure 18. Configuration Management is notified only after repeated errors and then a failure is declared only after a critical threshold is reached or self tests reveal a hard failure.

The first task of Configuration Management is to decide what terminal to charge with the error. While determination that an error occurred is primarily based on the receiver's status word (or lack of it), the error is normally attributed to the transmitting terminal. The receiver is charged with the error only in the case that he reports an error when neither the Master processor/BCIU nor the transmitting terminal detect any anomaly.

Two error counters are maintained by Configuration Management for each bus side of each terminal address. The history counter represents an accumulated count of errors since system startup. It is cleared only during startup or reconfiguration. The current error counter represents the number of errors tallied since the last successful execution of self-test.

These error counters are compared with two threshold values specified by the system designer for the terminal address under analysis. If neither threshold is exceeded, the procedure simply resumes the normal retry sequence. If the current error threshold is exceeded, terminal failure analysis is initiated to either clear or confirm the error condition. If the history count threshold is exceeded, the terminal is flagged as failed for this bus without any further analysis or expenditure of overhead.

When terminal failure analysis is invoked, it will cause self-tests, which are largely bus independent, to be run. If these tests and the subsequent BIT word analysis reveal a serious error condition, the entire device may be flagged as failed. Also, since terminal failure analysis can be a lengthy procedure, the system must protect itself from excessive testing. A device will be flagged as failed if terminal failure analysis is invoked for it twice in the same minor cycle.

Another condition under which an entire device is failed is that of a failure on one bus when the Master previously had been failed on the other bus.

When an entire device is flagged as failed, Configuration Management will perform one of several actions depending on the type of device. If a remote terminal is failed, Mission Application Software will be notified by way of the Subsystem Status Monitor, so that appropriate recovery or backup of the failed subsystem may be activated. Also, the bus messages to that terminal are "NO-OPed", e.g. the terminal declared as "down". Terminals that have been declared as "down" by configuration management are periodically polled. If the terminal recovers, self test of the terminal is initiated. If the terminal is declared as "up" the messages to that terminal are restored, and application software notified of the recovered device.

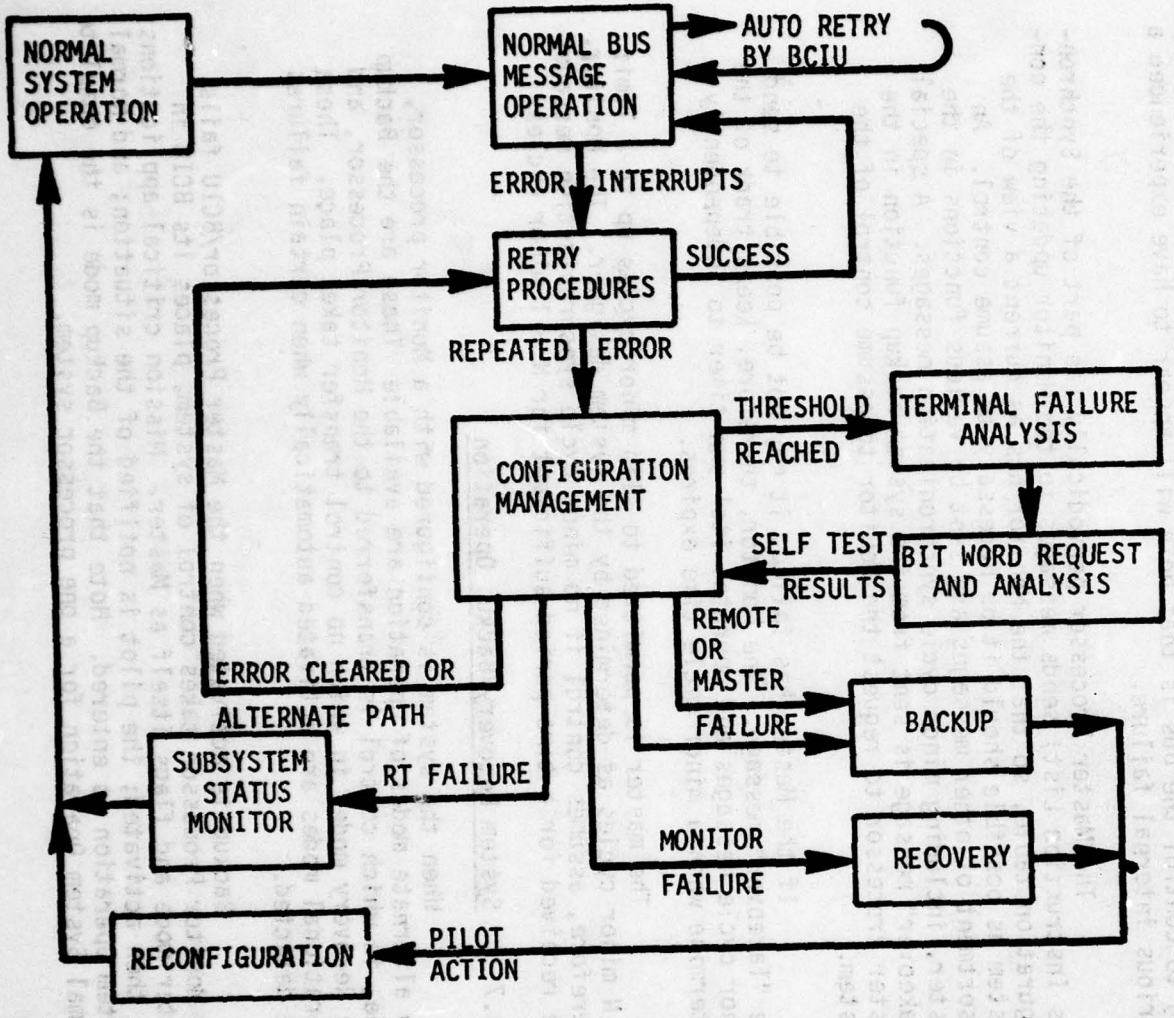


FIGURE 18. CONFIGURATION MANAGEMENT INTERACTIONS

4.1.6 Monitor Management

When a Monitor Processor is included in a configuration, it is loaded with a copy of the Master Executive and that subset of application tasks which are deemed to be mission critical. The Master Executive maintains an internal flag to indicate whether it is operating in Master or Monitor mode. The Monitor BCIU is operated in remote mode, since only one unit may control the bus at any one time. This is a critical point since the likely outcome of two units attempting to control the bus is that both will appear to have experienced a serious internal failure.

The Master Processor periodically (as part of the Synchronous Instruction List) sends messages to the Monitor updating the configuration table, so that the Monitor has as current a view of the system as possible should it be necessary to assume control. An assortment of other messages are sent by various functions in the Master, including minor cycle synchronization messages. A special "Takeover" message is sent from the system backup function in the Master Processor to request the Monitor to assume control of the system.

If the Master has failed, it may not be possible to send the "Takeover" message. The Monitor, therefore, keeps track of the minor cycle messages and operates its' own timer to independently determine when a minor cycle time expires.

The master is permitted to slip minor cycles up to a limit of N minor cycles as determined by the system designer. The Monitor, therefore, assumes control if no minor cycle synchronization messages are received for a time period sufficient for N + 1 minor cycles.

4.1.7 System Recovery/Backup Operation

When the system is configured with a Monitor processor, two alternate modes of operation are available These are the Backup mode, in which control is transferred to the Monitor Processor, and the Recovery mode, in which no control transfer takes place. These operational modes are activated automatically when certain failures are detected.

Backup is activated when the Master Processor/BCIU fails. The Monitor Processor takes control of system, places its BCIU in master mode and flags itself as Master. Mission critical applications are then activated; the pilot is notified of the situation; and Normal System Operation is entered. Note that the Backup mode is the complete Normal System Operation for a one processor system.

Backup is also activated if a remote processor/BCIU fails. This is necessary since interdependent mission application software modules are distributed across the Master and Remote Processors. The loss of any one of them, therefore, causes improper operation of the others.

Recovery is activated when a failure of the Monitor Processor is detected. The Master attempts to shut down the Monitor to prevent him from erroneously attempting to assume control of the bus. The pilot is advised of the situation and normal system operation resumed. Note that this is a very rapid operation and the complete system functions are maintained. The only loss is that Backup is no longer available as an alternate mode of operation.

As noted above, the pilot is notified when either Backup or Recovery is activated. Normally, the pilot will then execute the reconfiguration procedure at a future convenient time, as determined by mission requirements and other external factors.

4.1.8 System Reconfiguration Operation

When the pilot receives the indication that a processor/BCIU has failed, the system will have already transitioned itself to the Backup or Recovery mode of operation. He has the option of depressing the corresponding power enable button on the Processor Control Panel (PCP) to power down the failed unit and prevent it from interfering with the operational system elements. If the processor is left powered up, the system will attempt to determine if and when the unit recovers and advise the pilot a reconfiguration may be attempted.

The pilot selects a convenient time consistent with mission requirements and initiates the reconfiguration by pressing the START or LOAD button. This will cause an INFLIGHT RESTART as defined in section 4.1.1 and result in the system making maximum use of the operational resources.

4.2 DAIS Architecture Features

The DAIS architecture possesses specific characteristics which facilitate the adaptability of the system to various applications, as well as supporting effective incremental integration and test of the DAIS core elements. These key characteristics are discussed in the following sections.

Specific features in the DAIS system which allow adaptation of the core elements to a specific avionics system configuration are presented below.

4.2.1 Bus Devices

The basic message structure, data format and command-response protocol required for the multiplex system is described in Section 4.1.2. The message protocol is defined as that sequence of bus messages required to perform normal multiplex synchronous and asynchronous operations, and to support message error detection and recovery process. This definition of the command formats allow up to 32 different addressed devices to be implemented on the multiplex system. DAIS has further expanded this definition to allow up to four processors/BCIU, and then up to 28 remote terminal devices.

The system designer would define the specific hardware configuration (number of processors/BCIUs, number of RTs to interface with the avionic sensor and subsystems using the standard Interface Modules (IMs) or unique IMs if required). The system designer would then define the sensor inputs/outputs as data bus messages, and input this information to PALEFAC. PALEFAC would generate the Executive bus instruction list tables accordingly, which will direct the messages to the proper processor containing the application software (e.g. EQUIPS module) which would process the sensor/subsystem data.

The DAIS architecture is readily adaptable to various numbers of devices on the bus, and provides the tools to automatically generate and direct the messages to the proper application software in one of the processors.

4.2.2 Processor/Bus Controllers

The architecture provides the capability to expand or reduce the number of processors/BCIU for a specific application. The system designer would perform trade-offs in determining the number of processors/BCIU required considering mission requirements, reliability and availability.

The system designer would define the mission software application modules (inputs, outputs, and algorithms) based upon the mission requirements and avionic suite. The application programmer would develop the application modules using the J73/I language and mission software standards. Based upon the application execution times, memory size, data access, comsubs, and the bus message traffic, the system designer would then partition the application tasks among the processors using PALEFAC. PALEFAC would produce the tables for the master executive, and local executive required for run time execution of the application software in the specific configuration of processors/bus controllers.

4.2.3 Remote Terminal (RT)/Interface Modules (IMs)

The RT has a modular and programmable design to provide flexible partitioning of the data messages to the appropriate sensors/subsystems, and signal conditioning required by different numbers and mixes of subsystem signals. This is accomplished as follows:

The Timing and Control Unit (TCU) in the RT performs all of the timing, control, buffering, decoding and checking required to receive or transmit information from the data bus and transfer that information as outputs or inputs from the RT, via the Interface Modules (IM). The TCU contains a programmable device which controls the mapping of each data word in the RT. The interface between the TCU and IMs is standardized and contains the signals required to allow TCU to select the individual modules. Therefore, all IM slots can accept all IM types.

Each RT will interface with different numbers and mixes of sensor/subsystem signals for each specific system configuration. This is accomplished by inserting the proper number and type of interface modules into the IM slots in the RT housing and appropriate ROM programming in the TCU. If required, a special interface module can be designed to interface with an existing subsystem with a unique interface.

The functions of the RT can also be embedded in a sensor or subsystem, so that the interface of the sensor or subsystem is directly with the multiplex data bus. Functionally, the embedded RT would respond to commands received on either data buses the same as an RT.

4.2.4 Applicable Software

The DAIS System Architecture is designed to allow modular implementation of specific systems by using the required elements of the DAIS system, both hardware and software. Given that the set

of multiplex equipment has been chosen for interfacing to the external world (sensors and Controls and Displays equipment), the design of the processing system can begin. The Application Software is initially designed as if it will exist in one large virtual processor with as much memory space and execution time available as the sum of the flight processors to actually be used in the system. After the data interfaces have been defined to the outside world (multiplex bus messages), and the functional performance required of the Application Software has been identified, the layout of Compool Blocks and tasks can begin.

Compool Blocks are the data communication paths between the Application Software and the external world (over the multiplex bus), as well as between the Application Software tasks. Tasks are the processing elements in the Application Software which collectively perform the Avionics processing function. Tasks access the Compool Blocks through calls upon Executive services (such as READ and WRITE) and operate upon the contents of the COMPOOL BLOCK as local data for processing purposes.

After the functions of the Application System have been designed in terms of Tasks and Compool Blocks, and the application software has been tested on the Host Processor as the single virtual processor, partitioning of the Application System onto the flight processors begins. This partitioning is illustrated in Figures 19 and 20 for a two and three processor system configuration respectively.

This capability to easily partition the Application Software across a set of flight processors is a direct result of the use of the DAIS Executive to provide general I/O control for all communications on the multiplex bus (between processors as well as to the external world), and the use of PALEFAC to build the data base which is then used by the Executive to control system communications. The use of the Executive and PALEFAC results in an automatic partitioning of the Compool Blocks on to processors based upon the partitioning of Tasks, and which Tasks use which Compool Blocks. If a Compool Block is used for communications between tasks, and the tasks are split into two or more processors, the System will automatically generate a copy of the Compool Block at each processor, as well as generating the multiplex bus message to update all other copies when one copy is updated .

The system will also automatically generate the multiplex bus messages required to connect the Compool Blocks, which interface to the outside world, to the devices on the multiplex bus. All multiplex bus messages may be declared as synchronous with a period and phase, and if not so specified, the System will automatically generate asynchronous updates of the bus messages when a copy is updated.

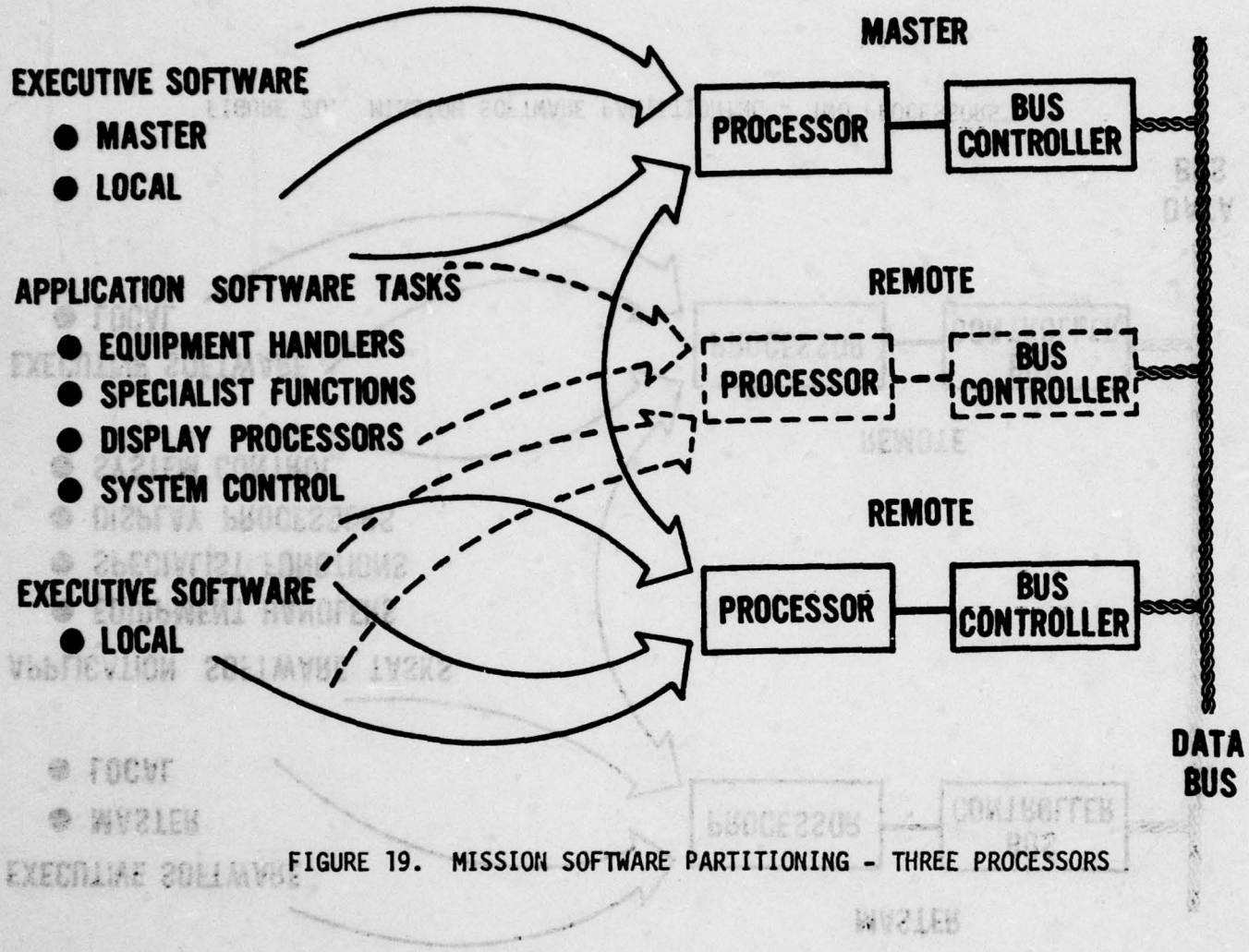


FIGURE 19. MISSION SOFTWARE PARTITIONING - THREE PROCESSORS

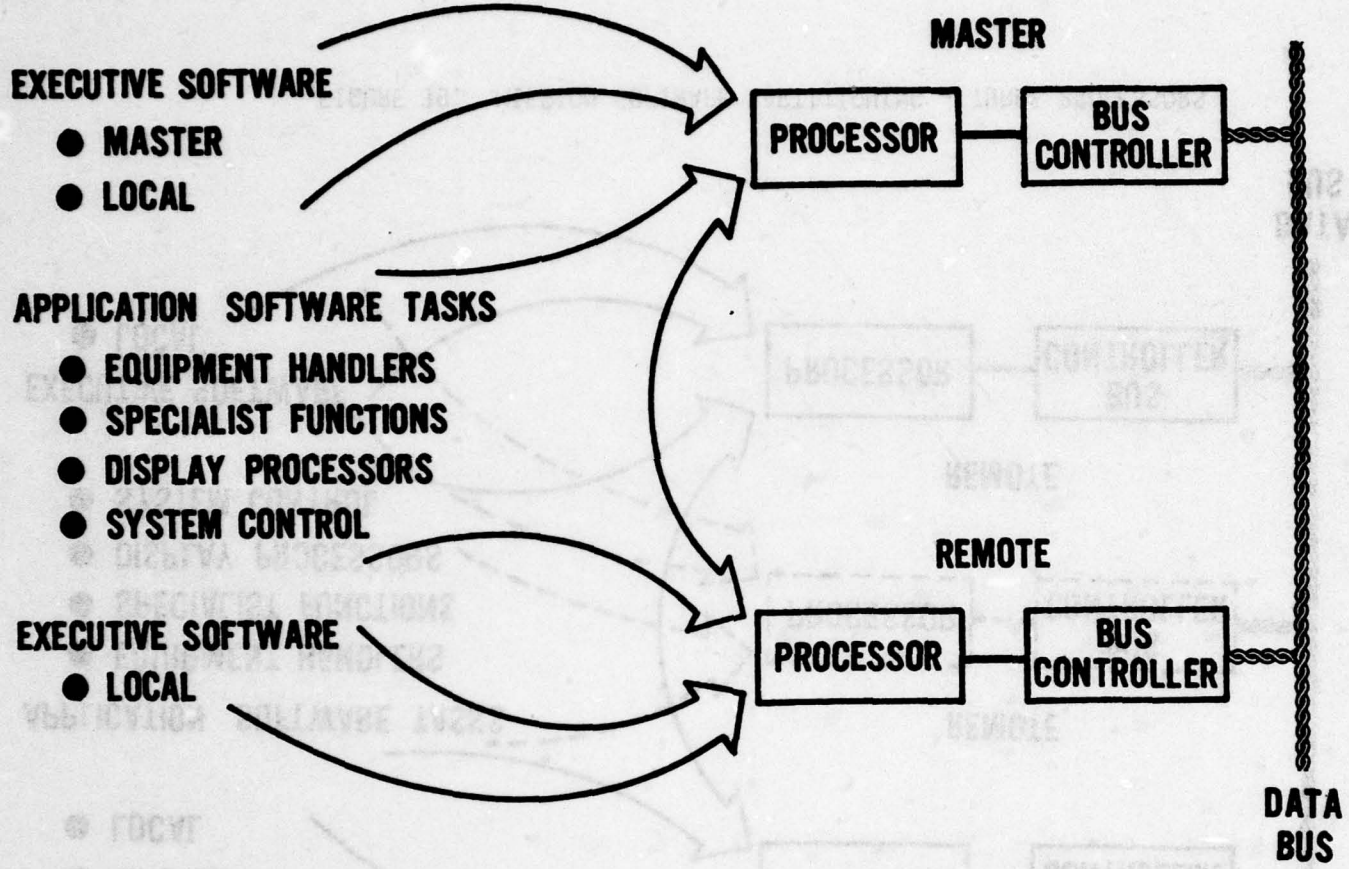


FIGURE 20. MISSION SOFTWARE PARTITIONING - TWO PROCESSORS

4.2.5 Interface "Standards"

4.2.5.1 DAIS/1553A Message Protocol

As discussed previously, MIL-STD-1553A defines the basic message structure and command-response protocol. In order to properly handle detected message errors and other DAIS system features, DAIS further expanded the definition of 1553A and defined a set of procedures, designated the system control procedures, that defines the mechanism to support bus operations and message error and terminal failure detection and recovery process. This composite definition of DAIS/1553A protocol establishes a "standard". If a sensor/subsystem device's bus interface conforms to this "standard", it would be fully compatible with the DAIS System Architecture.

4.2.5.2 Executive Application Software Services

As discussed previously, the DAIS Executive has been designed from the point of view of supplying a set of standard services to Application Software. These services are provided to the Application System through a set of software interfaces which are fully specified and are Application invariant. The result of standardization of the interface is that the Executive is truly Application independent and can be used as a modular piece of the DAIS System.

4.2.6 Portability

The DAIS Mission Software approach has, from the very beginning, been conceived of an easily retargetable system. The single most important feature that allows this capability is the development of more than 97% of the DAIS Executive Software in JOVIAL J73/I (the only routine written in assembly language are processor dependent I/O operations, register manipulation operations and lowest level interrupt handlers). All of the DAIS Application Software is implemented in JOVIAL J73/I. A striking demonstration of the portability of the DAIS Mission Software is the fact that it executes both on the DAIS processor and on the DEC-System-10, the host processor facility. In fact, standalone module testing and initial subsystem integration testing of Application Software is performed on the Host Processor. This is possible since the JOVIAL J73/I Compiler has a code generator for both the DEC-System-10 and the DAIS processor, which points out the ease of retargetability of the DAIS Software System.

4.2.7 Redundancy

DAIS provides redundancy in the system architecture which can be employed to provide backup and recovery to complete mission functions in spite of hardware failure. The areas which can be used for redundancy are:

a. Dual redundant data bus including redundant bus interface modules in the BCIU and RT.

b. System can employ dual redundant RTs for a specific subsystem.

c. Multifunction keyboard and associated Data Entry Keyboard can be used as a backup to the Integrated Multifunction Keyboard and associated Data Entry Keyboard.

d. Raster displays can serve as backup to each other.

e. A processor/BCIU can be designated as monitor, serve as backup to the master processor/BCIU.

5.0 Support Facility

The purpose of the AVSAIL support facility is to test and evaluate the DAIS architecture and a representative set of core elements. The support facility provides a real time simulation of a military aircraft performing an operational mission. The simulation generates the interface signals between the simulated aircraft sensor suite and the DAIS system so that the avionic equipment is subjected to a data signal environment which is nearly identical to actual flight. A simulated cockpit is included as part of the simulation for realistic evaluation of the avionic system.

The support facility for DAIS consists of a Software Test Stand (STS) and an Integrated Test Bed (ITB). The Software Test Stand provides a capability to test and check out the mission software resident in the DAIS flight processors. The Integrated Test Bed provides the capability to test the DAIS core elements which include the mission software, the DAIS processors, the multiplex bus system and the cockpit control and displays. Both the STS and ITB utilize a complex of digital computers consisting of a Digital Equipment Corporation (DEC) DEC-10 and a number of DEC PCP 11 series computers.

Simulation software resident in the DEC-10 was developed to provide real time simulation of a military aircraft in an operational environment including the aircraft dynamics, the aircraft sensors and weapon targets. The simulation is driven from the cockpit by an operator acting as a "pilot". The simulated cockpit is equipped with control and displays so that the various modes of a mission may be "flown" by the "pilot" with an out-the-window background scene.

The support hardware and software provides interfaces between the DAIS elements under test and the DEC-10 host simulation software. The support facility provides the capability to set up a test, conduct a simulation and record data from both the DAIS elements and the simulation. During a simulation, sufficient test data is displayed in real-time to indicate that the system is operating satisfactorily to provide presentation of test results for observing systems performance.

The support facility is comprised of:

1. STS Support Hardware
2. STS Support Software and PDP-11 Processors
3. ITB Support Hardware
4. ITB Support Software and PDP-11 Processors
5. Host Simulation Processor, DEC-10
6. Simulation Software
7. System Test Software
8. Picture System
9. Picture System Software

5.1 System Configuration

The support facility functional diagram is shown in Figure 21 and consists of STS & ITB hardware and software both sharing the Host Simulation Processor and the DAIS cockpit. Support software is resident in the DEC-10 and in the PDP-11 Processors.

5.1.1 Integrated Test Bed (ITB)

The Integrated Test Bed (ITB) block diagram is shown in Figure 22 and is comprised of the support facility and DAIS core ele-elements. The support facility simulates all the equipment and environment external to the DAIS processor with the resident mission software, DAIS multiplex system, and DAIS controls and displays. The ITB support facility is comprised of support hardware, support software resident in the PDP-11 processor, the simulation software resident in the DEC-10 host simulation processor, the DAIS simulated cockpit, and an out-the-window picture system.

The support hardware provides the interfaces between the simulation and the DAIS core elements. The interface includes: (1) the multiplex bus messages which drive the mission software, (2) the performance monitoring of the multiplex bus traffic and the internal operation of the mission software in the DAIS processors, (3) the cockpit controls and backup instruments, and (4) the out-the-window picture.

The support software controls and manages the support hardware and provides the means to link the DEC-10 simulation software with the DAIS elements. The support software also provides the performance monitor and control functions which are related to test set-up, control and data collection for post-test analysis.

The host simulation processor is a commercial DEC-10 complex. The simulation software generates the real world environment external to the DAIS processors in real time. The simulation is driven by the DAIS cockpit flight controls so that an operator can "fly" the simulation.

An out-the-window picture system provides a symbolic background scene outside of the cockpit so that the "pilot" operator will experience the visual orientation of flight with respect to IP's, targets and runways. The picture system also provides the option for displaying the HUD symbology overlayed on the background scene.

A pictorial representation of ITB is shown in Figure 23.

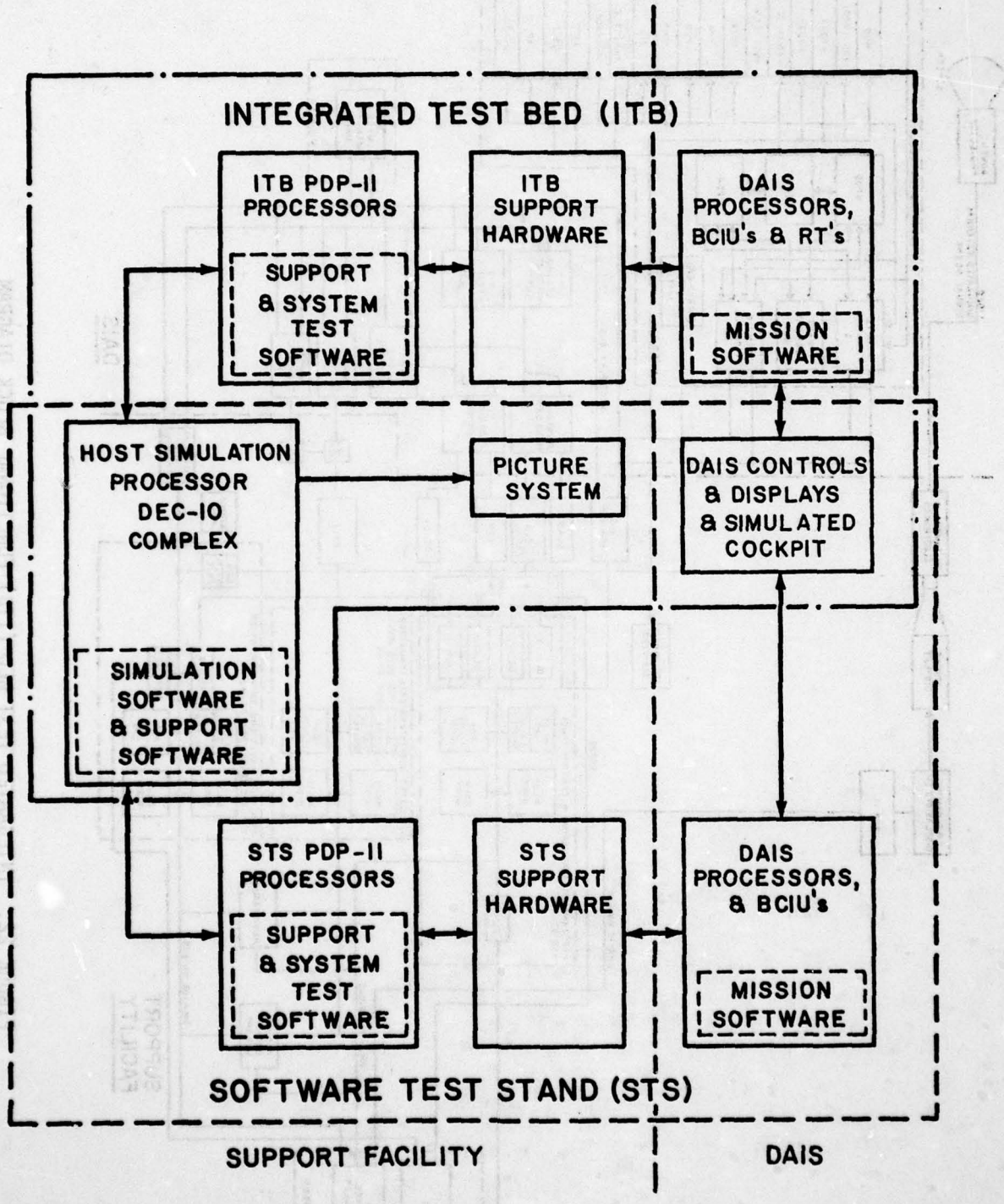
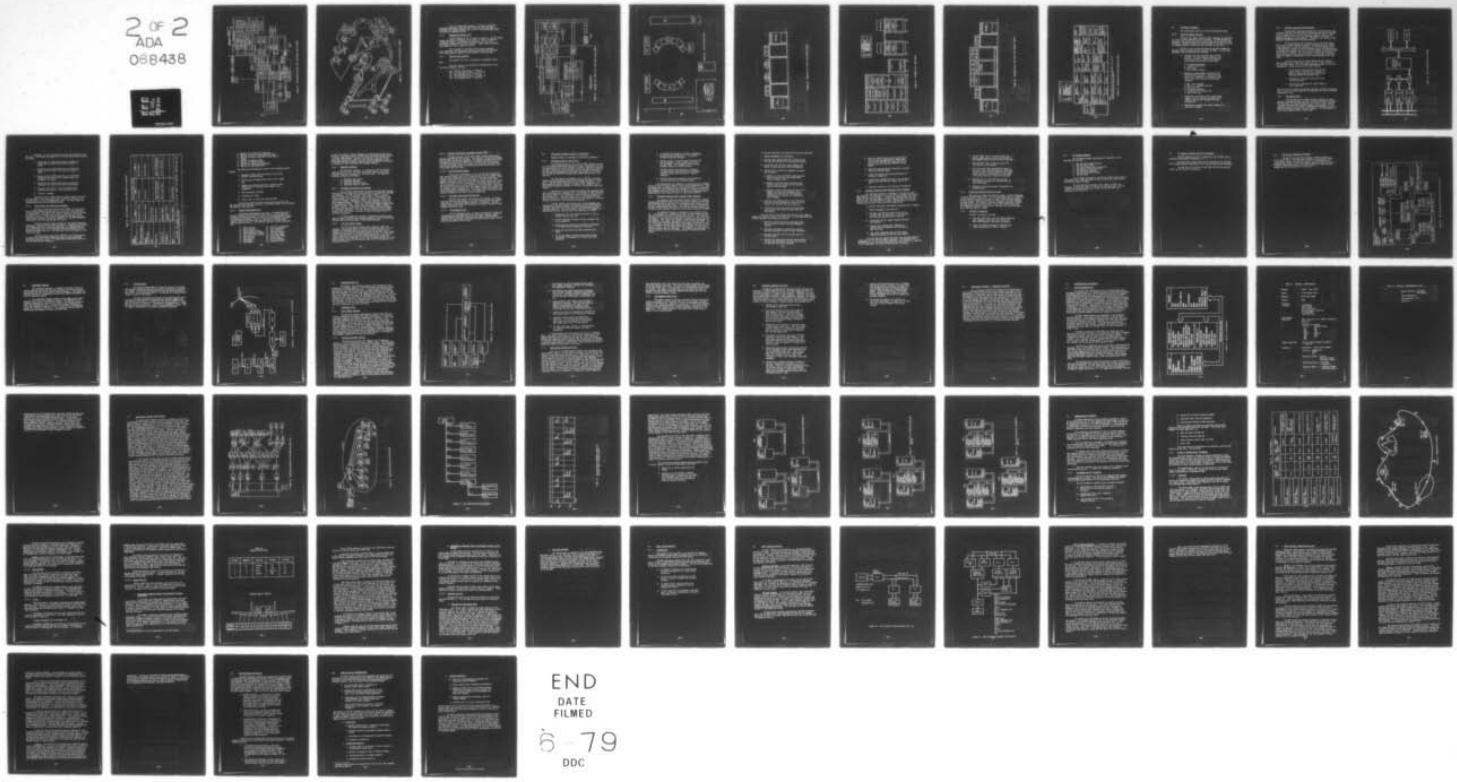


FIGURE 21. SUPPORT FACILITY FUNCTIONAL DIAGRAM

AD-A068 438 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/3
DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEM--ETC(U)
MAR 79 R C MASON, T R PRICE, B A RICH F33615-75-C-1178
UNCLASSIFIED AFAL-TR-79-1027 NL

2 OF 2
ADA
068438



END
DATE
FILMED
5-79
DDC

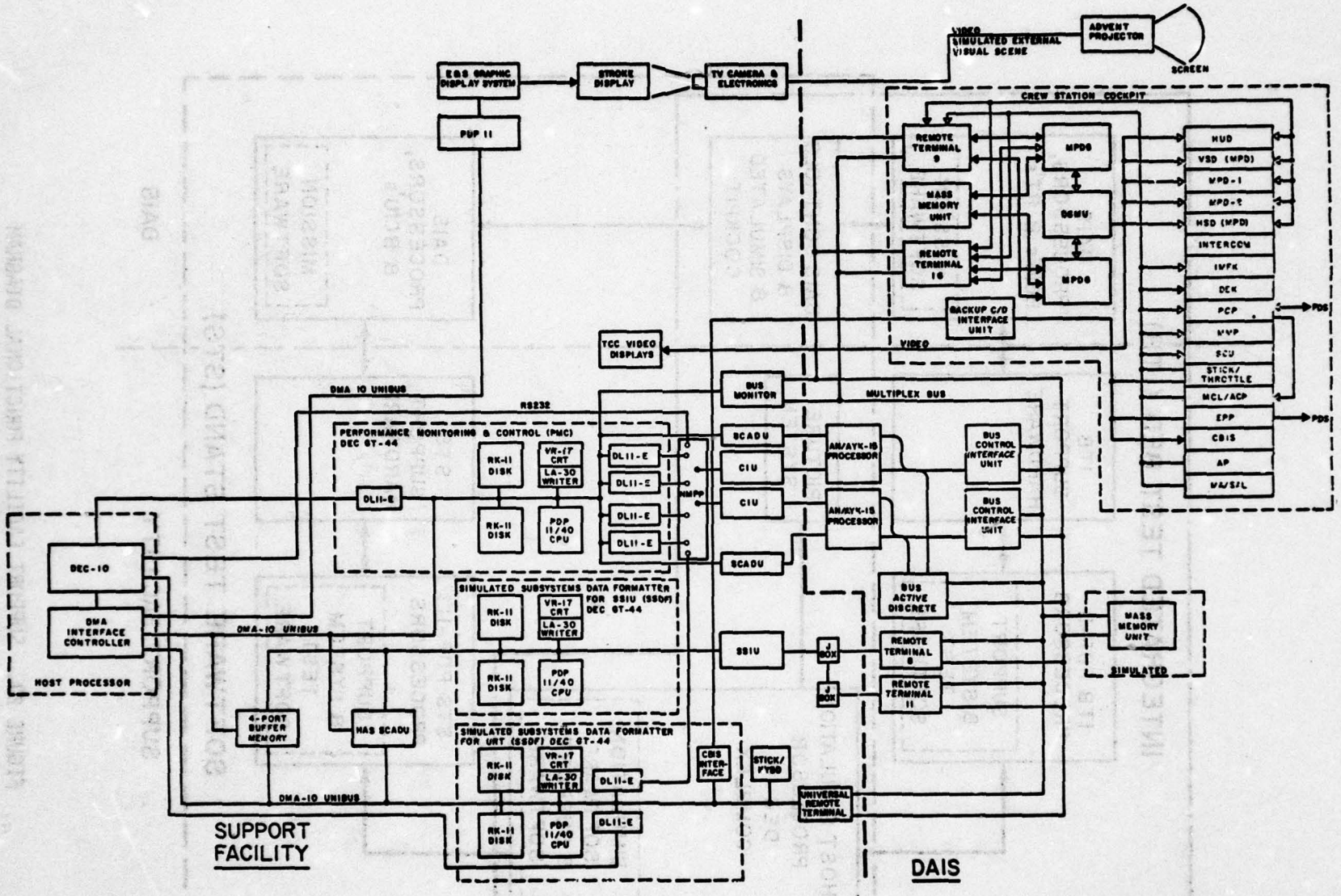


FIGURE 22. INTEGRATED TEST BED (ITB) FUNCTIONAL BLOCK DIAGRAM

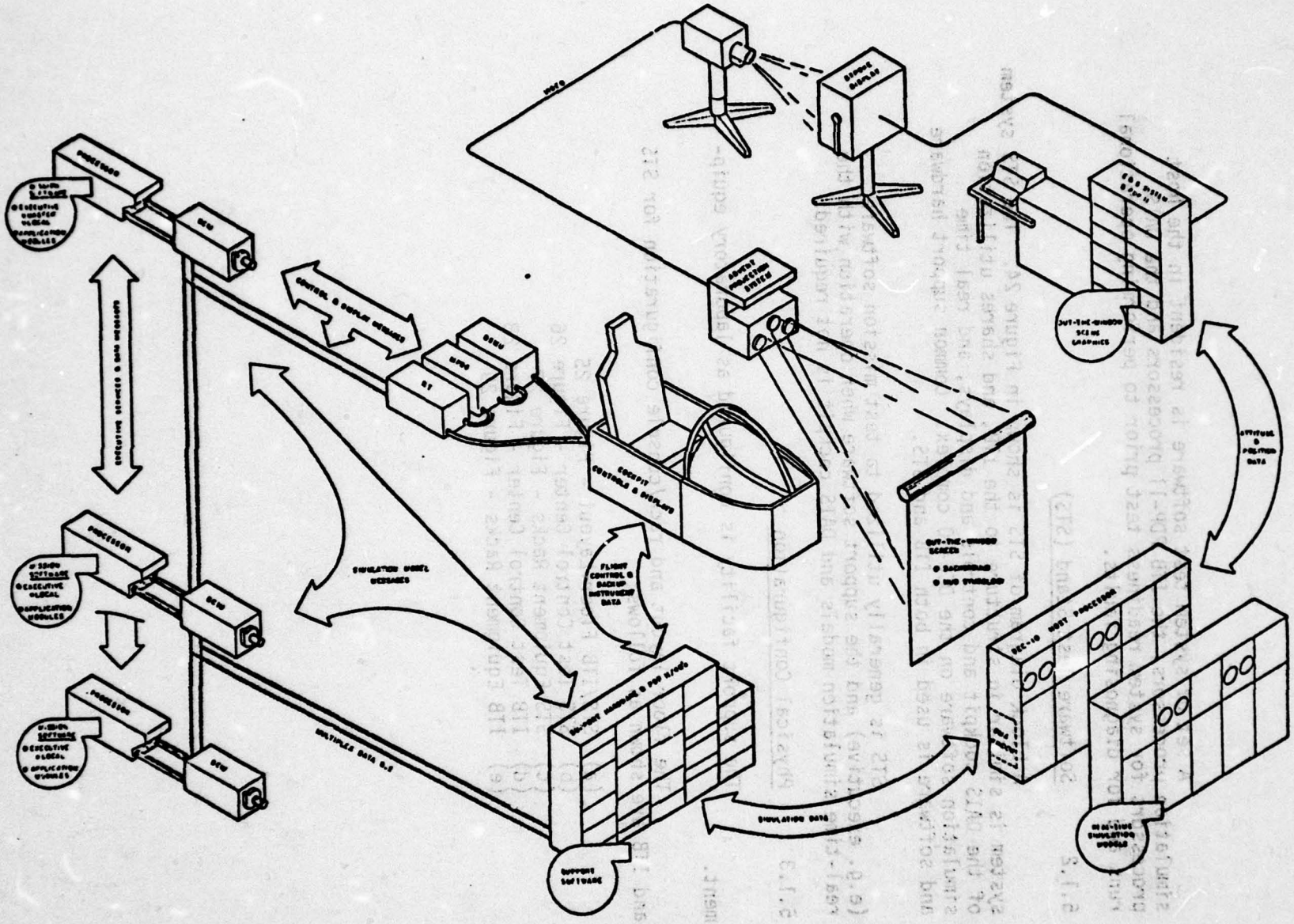


FIGURE 23. INTEGRATED TEST BED PICTORIAL DIAGRAM

A set of system test software is resident in the host simulation processors, the ITB PDP-11 processors and the DAIS processors for system readiness test prior to performing operational runs and for diagnosing faults.

5.1.2 Software Test Stand (STS)

A block diagram of STS is shown in Figure 24. The STS system system is similar in structure to the ITB, and shares utilization of the DAIS cockpit and controls and displays, and real time simulation software on the DEC-10 complex. Common support hardware and software is used in both ITB and STS.

STS is generally utilized to test mission software (e.g. executive) and the support software when operation with the real-time simulation models and DAIS cockpit is not required.

5.1.3 Physical Configuration

The support facility is configured as laboratory equipment.

The floor layout and rack/console configuration for STS and ITB are shown as follows:

(a) STS/ITB Floor Layout - Figure 25
(b) STS Test Control Center - Figure 26
(c) STS Equipment Racks - Figure 27
(d) ITB Test Control Center - Figure 28
(e) ITB Equipment Racks - Figure 29

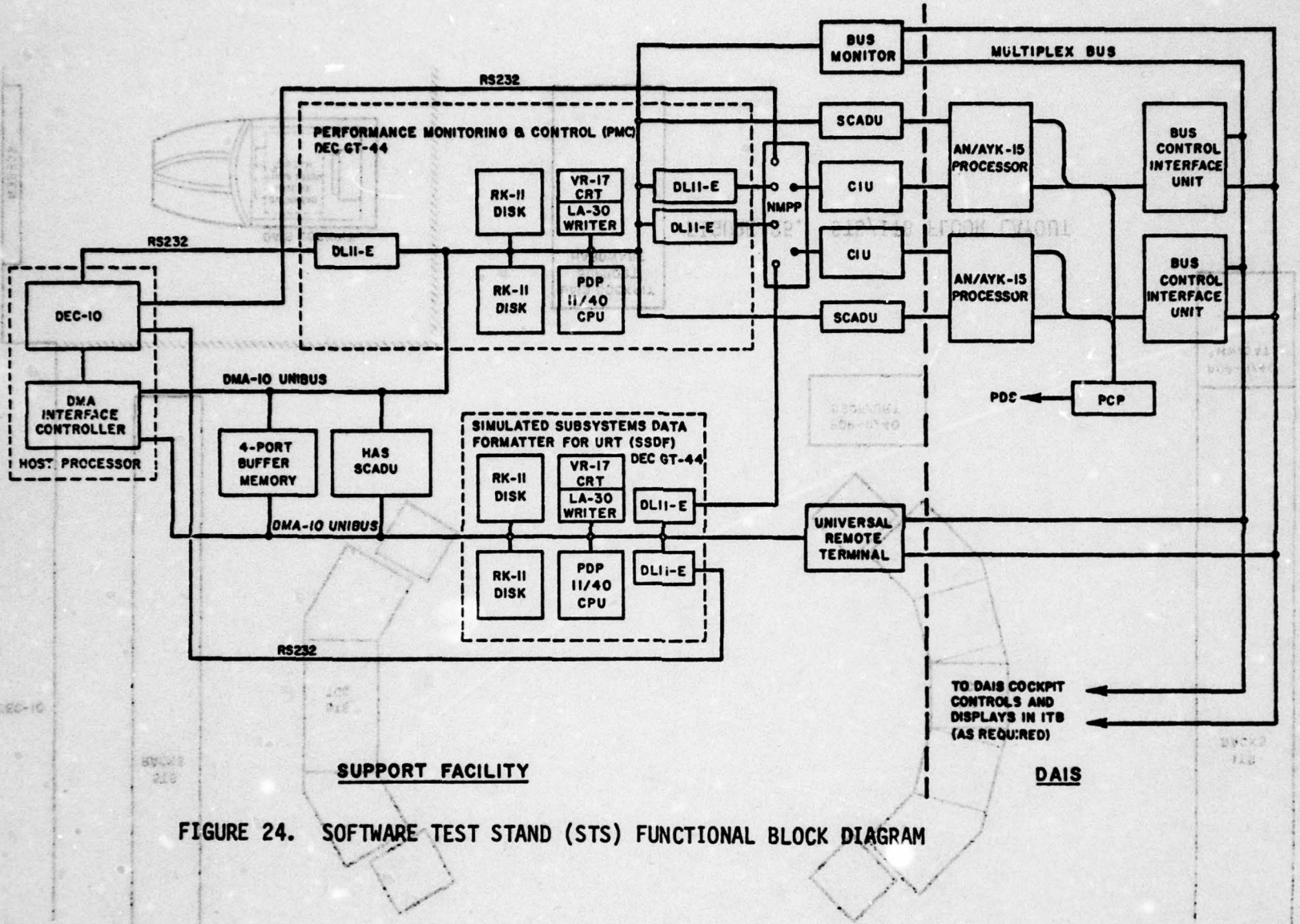


FIGURE 24. SOFTWARE TEST STAND (STS) FUNCTIONAL BLOCK DIAGRAM

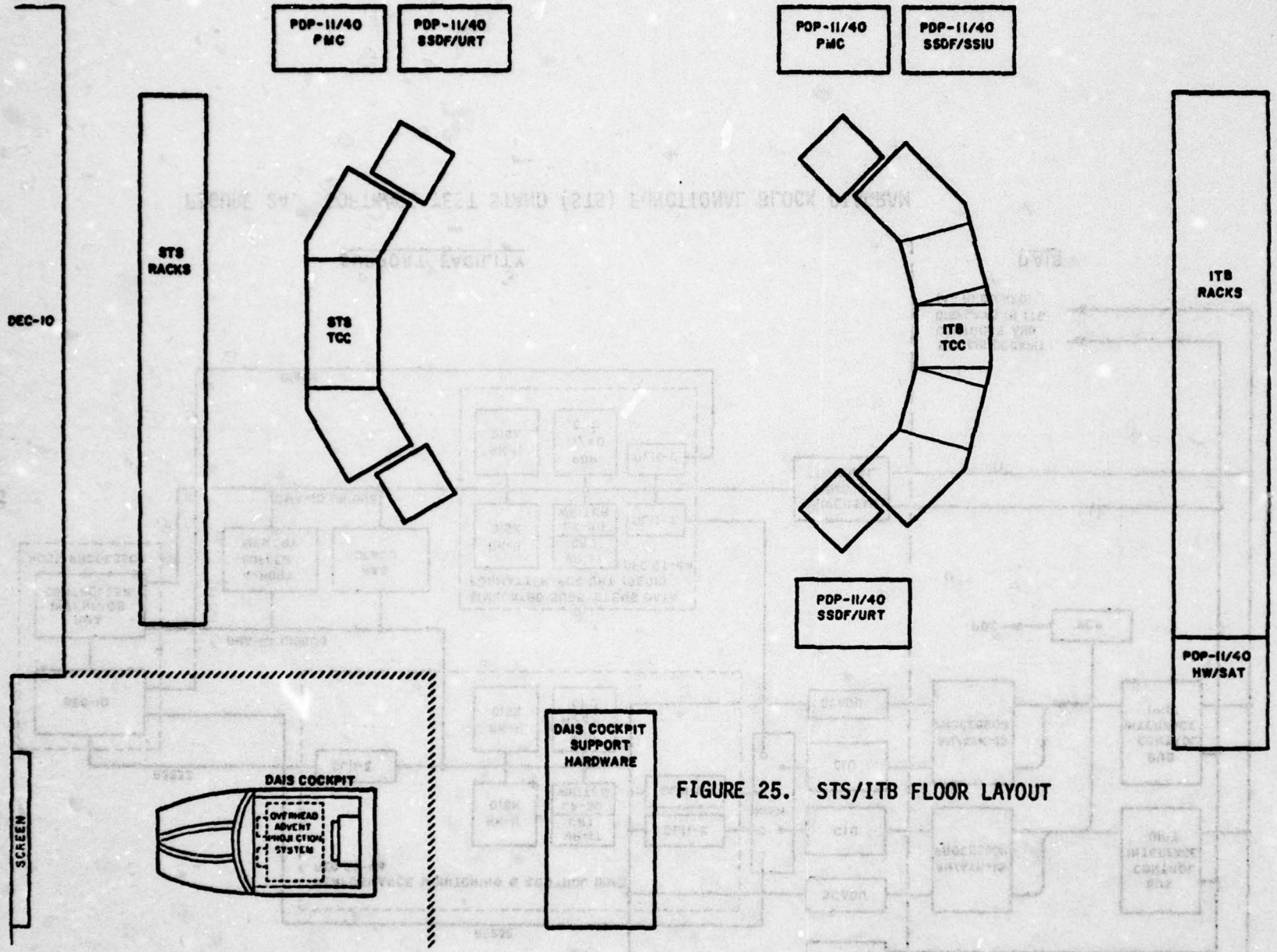


FIGURE 25. STS/ITB FLOOR LAYOUT

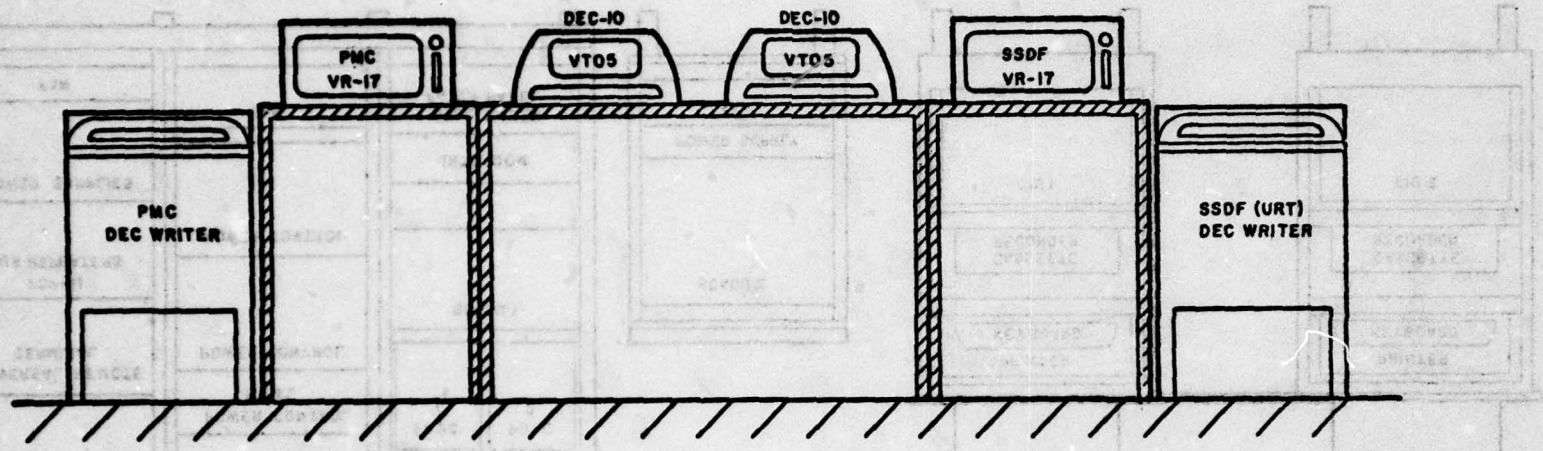


FIGURE 26. SOFTWARE TEST STAND TEST CONTROL CENTER

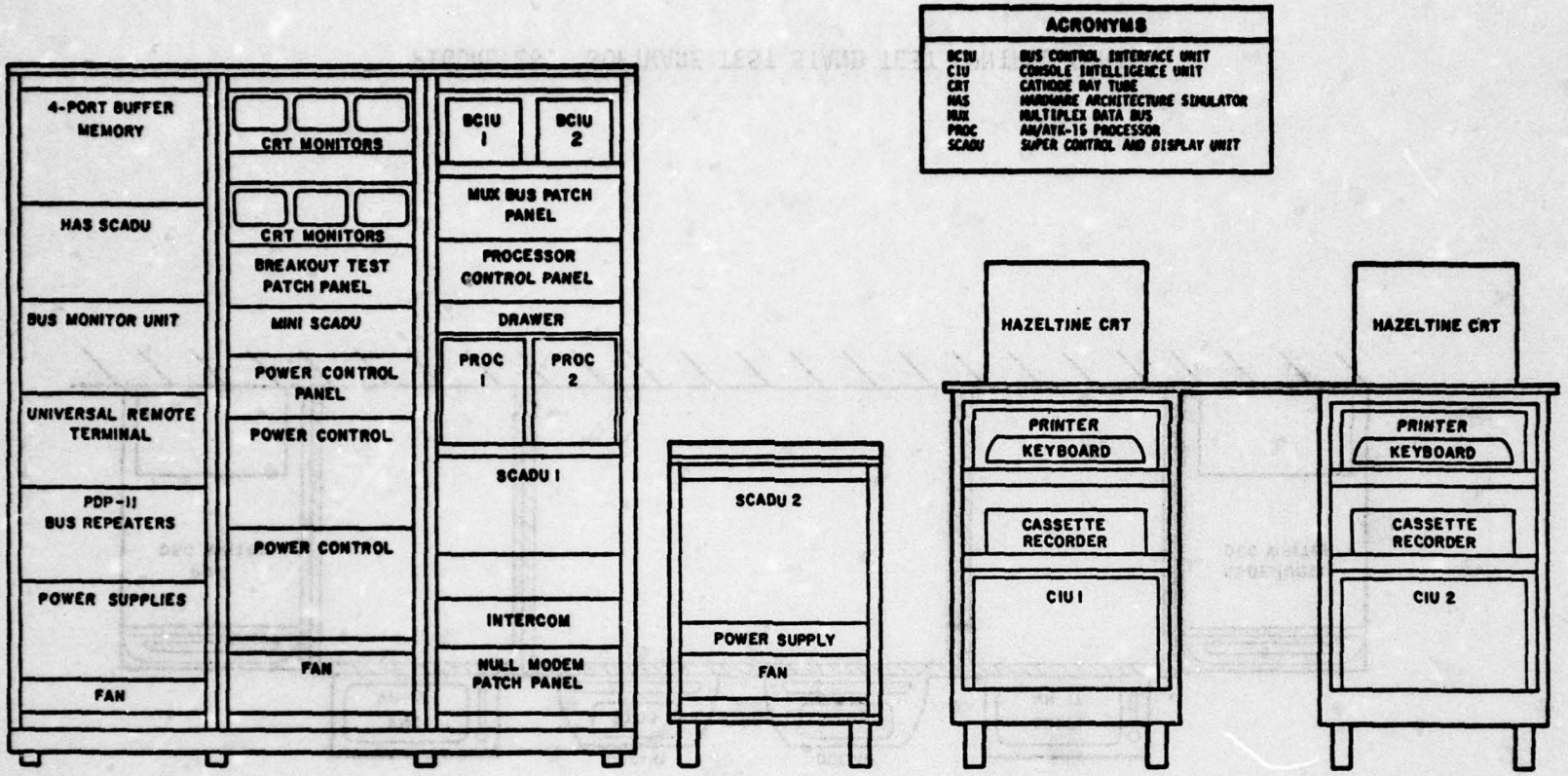


FIGURE 27. SOFTWARE TEST STAND RACKS

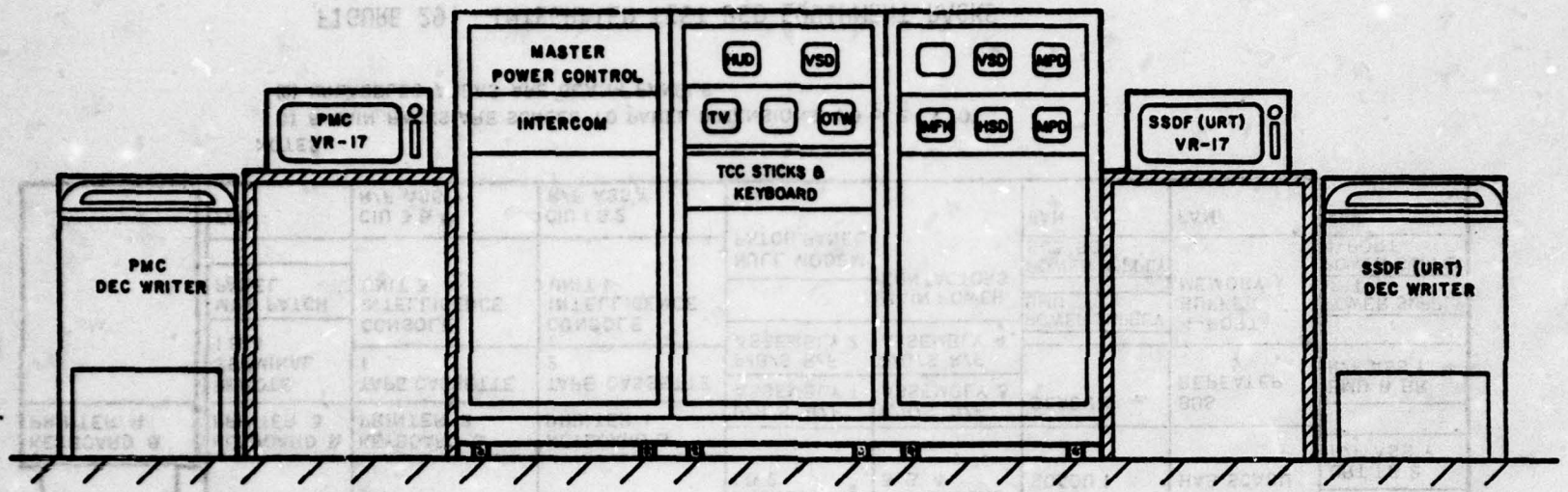


FIGURE 28. INTEGRATED TEST BED (ITB) TEST CONTROL CENTER

ACRONYMS	
BCIU	BUS CONTROL INTERFACE UNIT
BMU	BUS MONITOR UNIT
BR	BUS REPEATER
CIU	CONSOLE INTELLIGENCE UNIT
HAS	HARDWARE ARCHITECTURE SIMULATOR
MUX	MULTIPLEX DATA BUS
P/B/S	PROCESSOR/BCIU/SCADU
R/F ASS'Y	RELAY/FILTER ASSEMBLY
SCADU	SUPER CONTROL & DISPLAY UNIT
URT	UNIVERSAL REMOTE TERMINAL

	CONSOLE INTELLIGENCE UNIT 4	CONSOLE INTELLIGENCE UNIT 2	FAN	FAN	MASTER POWER PNL	POWER PANEL 1	POWER PANEL 2
			BCIU's 1 & 2	BCIU's 3 & 4	EMERG. STOP	POWER PANEL 3	
REMOTE TERMINAL R/F ASSEMBLY					PROCESSOR CONTROL PNL	SCOPE	LOGIC DRAWER
			MUX PATCH PANEL	MUX PATCH PANEL			POWER SUPPLY
HAZELTINE TERMINAL 4 CRT	HAZELTINE 2000 TERMINAL 3 CRT	HAZELTINE 2000 TERMINAL 2 CRT	STORAGE DWR		BUS MONITOR UNIT	UNIVERSAL REMOTE TERMINAL 1	HAS & 4-PORT R/F ASS'Y
		HAZELTINE 2000 TERMINAL 1 CRT	PROCESSORS 1 & 2	PROCESSORS 3 & 4	SCADU 1	HAS SCADU	
KEYBOARD & PRINTER 4	KEYBOARD & PRINTER 3	KEYBOARD & PRINTER 2					URT 1 & 2 R/F ASS'Y
		KEYBOARD & PRINTER 1	P/B/S R/F ASSEMBLY 1	P/B/S R/F ASSEMBLY 3	SCADU 2	BUS REPEATEP	
	REMOTE TERMINAL 1 & 4	TAPE CASSETTE 1	P/B/S R/F ASSEMBLY 2	P/B/S R/F ASSEMBLY 4			BMU & BR R/F ASS'Y
		TAPE CASSETTE 2			POWER SUPPLY BMU	4-PORT BUFFER MEMORY	
	MUX PATCH PANEL	CONSOLE INTELLIGENCE UNIT 3		MAIN POWER CONTACTORS			POWER SUPPLY URT 1
		CONSOLE INTELLIGENCE UNIT 1	NULL MODEM PATCH PANEL		POWER SUPPLY SC3 SC4		POWER SUPPLY 4-PORT
	FAN	CIU 3 & 4 R/F ASS'Y			FAN	FAN	FAN
		CIU 1 & 2 R/F ASS'Y					

NOTES:

(1) 8 MAIN RACKS ARE SCALED TO PANEL DIMENSIONS (19 or 24 x 70)

(2) UNLABELED AREAS ARE BLANK PANELS

FIGURE 29. INTEGRATED TEST BED EQUIPMENT RACKS

5.2 ITB Support Hardware

The ITB hardware consists of the following subsystems:

5.2.1 Universal Remote Terminal

The Universal Remote Terminal (URT) simulates the operation of a set of remote terminals (32 RTs maximum). The URT receives or transmits data between the DAIS multiplex data bus and the PDP 11/40 processor. The URT provides the same interface to the DAIS multiplex as a Remote Terminal, except the URT simulates up to 32 RTs.

The URT is set up and controlled by the SSDF software in the PDP 11/40 by loading the URT registers and RAMs. In real time operation, the URT performs the following:

1. Transfers the DAIS multiplex data to/from the PDP 11 for each message operation (controller-to-terminal, terminal-to-terminal, and terminal-to-controller) based upon:

   - RT address/subaddress
   - Transmit/receive bit
   - Word count
   - PDP 11 memory address

2. Responds to mode commands received on the DAIS multiplex data bus. The URT has the capability to generate an interrupt to the PDP 11 upon receiving the following mode commands:

   - MTU 1 or 2 shutdown
   - MTU 1 or 2 shutdown override
   - Reset bit word
   - Initialize terminal
   - Initiate serial channel I/O
   - Minor cycle sync

3. Capability to set/reset bits in the serial channel activity register and module error register, and the BIT register for each simulated RT.

4. Capability to inhibit the status response to any receive command.

5.2.2 Simulated Subsystem Interface Unit

The Simulated Subsystem Interface Unit (SSIU) provides the interface, control, and data transfer functions required to interface a set of remote terminals (RTs) to a PDP 11 unibus. Each RT interfaces to the SSIU via a Facility Interface Module (FIM) which is specially designed for this purpose. Figure 30 is an interface block diagram illustrating the SSIU interconnection.

The SSIU provides the interface for the simulated sensor suite that in actual operation would be interfaced by up to sixteen fully populated RTs with any complement of Interface Modules (IMs). A FIM occupies a single RT slot and responds to any slot addressed by an address which defines the IM slot and IM channel for which the data were actually intended. The SSIU uses this address as a pointer to store and retrieve data from unibus memory in preselected locations which are mapped to correspond to the RT configurations being simulated.

The SSIU uses a PDP 11 unibus memory that can support a Direct Memory Access (DMA) data rate of at least 1 MHz for each 16 bit word and operate in a burst mode (dedicated unibus) for up to 34 microseconds. Other features of the SSIU include:

1. At the end of each data block written into unibus memory, transferring a pointer by DMA to a preselected location defining the starting address of the data block;

2. Providing an interrupt at the end of preselected messages;

3. Providing an interrupt for several types of error conditions.

Set up of the SSIU memory and control registers, and monitor and control of the SSIU during operation are provided via unibus Programmed Input/Output (PIO).

5.2.3 Bus Monitor Unit

The Bus Monitor Unit (BMU) receives, records and breakpoints on selected messages received on the DAIS multiplex data bus. The BMU stores the received messages into the PDP 11/40. The BMU interfaces with the DAIS multiplex data bus and monitors for data messages (controller-to-terminal, terminal-to-controller, and terminal-to-terminal messages) and mode commands. The BMU operates in either the automatic mode or manual mode.

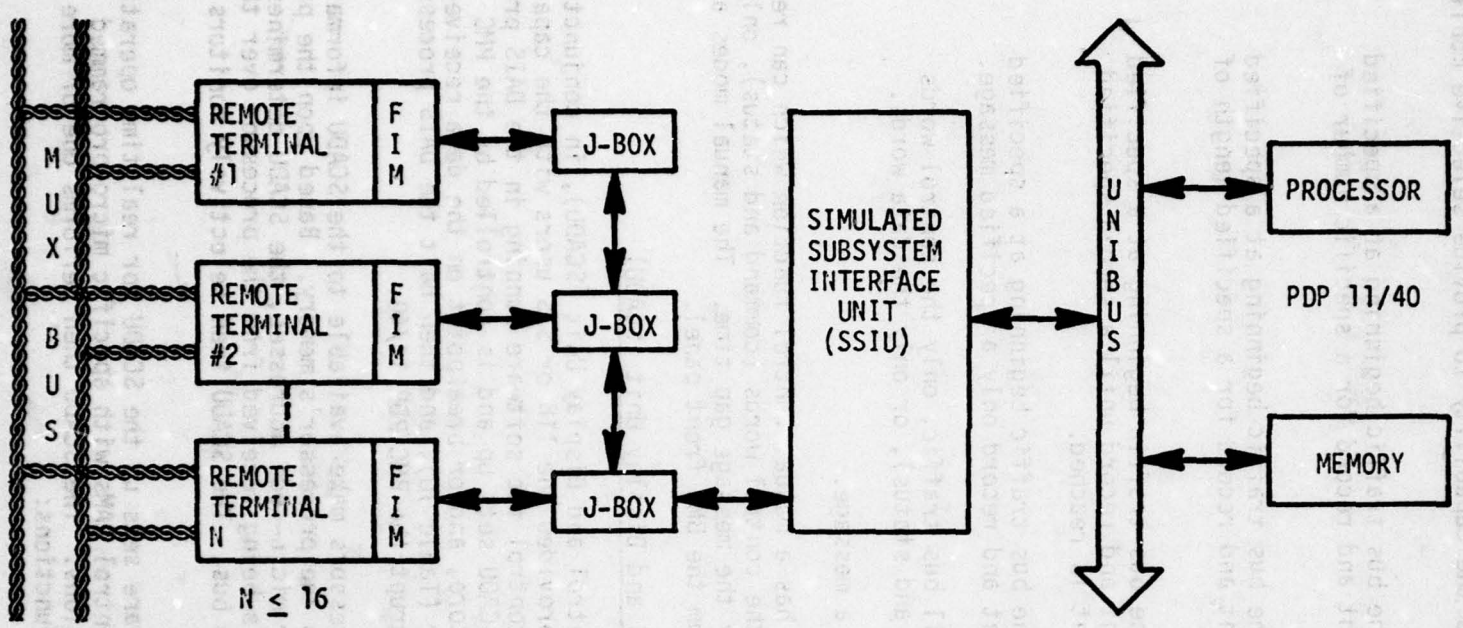


FIGURE 30. SSIU INTERFACE BLOCK DIAGRAM

The BMU is set up and controlled by the PMC software in the PDP 11/40. The BMU has the capability to provide selective monitoring as follows:

1. Record the bus traffic beginning at a specified breakpoint and record for a specific number of words.

2. Record the bus traffic beginning at a specified breakpoint and record for a specified length of time.

3. Record the bus traffic beginning at a specified breakpoint and record until a second specified breakpoint is reached.

4. Record the bus traffic beginning at a specified breakpoint and record only a specified message.

5. Record all bus traffic, only the control words (command and status), or only the data words.

6. Transmit a message.

The BMU also has a manual control function which can record all bus traffic, only the control words (command and status), only the data words, or only the message gap time. The manual modes are controlled directly from the BMU front panel.

5.2.4 Super Control and Display Unit (SCADU)

The Super Control and Display Unit (SCADU), in conjunction with the PMC software provides the ITB or STS users with the capability to monitor and control the software running in the DAIS processor (AN/AYK-15). The SCADU sets up and is controlled by the PMC software to collect, store, and/or breakpoint on the data received from the DAIS processor (Table 10); and then halt the DAIS processors and/or BCIUs, and interrupt the PMC PDP 11/40.

The DAIS processors make available to the SCADU information accessed or stored into the processor's memory. Based upon the processor's memory controller micro-code addresses, the SCADU determines which data (Table 10) is being received from the processor over the parallel multiplex data bus. The SCADU then selectively monitors and stores this data.

The PMC software sets up the SCADU for real time operation by loading the SCADU control RAMs with specific micro-programmed monitor and control actions. The SCADU then performs one or more of the following monitor functions:

TABLE 10. DATA AVAILABLE TO SCADU FROM AN/AYK-15 PROCESSOR

SCADU FUNCTION	SCADU DATA BUFFER CONTENTS		
	WORD A	WORD B	WORD C
IC (Instruction Address)	Instruction Address	Instruction	Operand Address
OPCODE (Instruction)	Instruction Address	Instruction	Operand Address
JUMPS	Operand Address		
STORES (1)	Operand Address	Most Significant Word (MSW) or Fixed Value	Least Significant Word (LSW) or Fixed Value
ACCESSES (1)	Operand Address	MSW or Fixed Value	LSW or Fixed Value
INTERRUPTS	Interrupt Address (20-3F)		
DMA's	Operand Address	Operand Value	

(1) NOTE: The contents of Word A, B, or C for the "STORES" and "ACCESSES" functions may vary depending on the instruction executed.

1. Monitor all instructions addresses.
2. Monitor a specific instruction (Op-code).
3. Monitor all jump instructions which cause a branch.
4. Monitor all memory stores.
5. Monitor all memory accesses.
6. Monitor all interrupt occurrences.
7. Monitor all DMA occurrences.

The SCADU also performs one or more of the following control actions:

1. Breakpoint (halt) upon occurrence of one of the specific functions above.

2. Log the data in SCADU buffer (trace) with a time tag.

3. Compare the function with user's specific value (fixed point or floating point numbers) and breakpoint.

4. Halt processors and BCIUs.

5. Interrupt PDP 11/40.

6. Start, stop, or reset the time tag timer.

After the appropriate control actions have occurred, the PMC software reads the data collected in the SCADU buffer, and reinitiates the same or a new operation.

5.2.5 Console Intelligence Unit

The Console Intelligence Unit (CIU), in conjunction with a RS-232 interface with the PMC PDP 11/40, DEC-10, and the Hazeltine terminals, provides the means to control, and load/change DAIS processor registers and memory. The CIU is supplied by the DAIS processor vendor to support the debug and test the mission software under non-real time conditions. The functions the user is able to perform are:

1. Clear Processor
2. Halt Processor
3. Toggle Memory Protect
4. Read Instruction Counter
5. Load Instruction Counter
6. Read Register
7. Load Register
8. Read Memory
9. Load Memory
10. Clear all Breakpoints
11. Set a Breakpoint
12. Step Processor
13. Execute Processor
14. Load Tape Image
15. Write Tape Image
16. Stutter Mode
17. DEC to Cassette
18. Cassette to DEC

In the "Local Mode", console commands are issued from the Hazeltine keyboard. When under control of the DEC-10 or PMC PDP 11/40, the CIU is in the "Remote Mode". The DEC-10 or PMC PDP 11/40 functionally interfaces with the CIU similar to the Hazeltine terminals. This allows the Hazeltine terminal functions to be performed at the DEC-10 for loading, or PDP 11/40 under PMC software control.

5.2.6 Hazeltine Terminals

The Hazeltine terminals, in conjunction with the Console Intelligence Unit (CIU), provides interactive control of the DAIS processor registers and memory as specified above. The Hazeltine terminals include the following:

1. Hazeltine CRT Display
2. Hazeltine Keyboard
3. Hazeltine Thermal Printer
4. Hazeltine Cassette Tape Unit

5.2.7 ITB Power Distribution and Control

The ITB Power Distribution and Control System (PDS) provides simulated aircraft electrical system control of the power for the DAIS core hardware. The PDS simulates three power sources: battery, master AC generator, and Ram-Air Turbine (RAT). The output of the simulated power sources provides power to two sets (AC and DC) electrical bus systems: emergency, primary, and secondary. Each DAIS core hardware element is capable of simulated operation off any one of the electrical buses. Also, each individual hardware element have an on/off control. This permits simulated power control or power failure of any one of the power sources, simulated electrical buses, or individual equipments. The PDS includes provisions for software control via a PDP 11/40 of the power sources, electrical buses, or individual equipments.

The PDS system also controls, distributes and filters the power to each of the DAIS core hardware elements. The PDS system also controls and distributes power to the support hardware.

5.2.8 ITB Test Control Center

The ITB Test Control Center (TCC), shown in Figure 28, provides a centralized point for control of the entire ITB. The TCC contains the terminals which can control the PMC and SSDF PDP 11 processors, and the DEC-10 simulation models via the PMC. Both system checkout and operation can be controlled from the TCC. The TCC contains a cockpit stick and throttle station which may replace the cockpit and allow an operator to "fly" the simulation at the TCC. The DAIS displays and the out-the-window scene are also displayed on CRT monitors at the TCC.

5.2.9 Controls and Backup Instruments System (CBIS)

The CBIS consists of the DAIS cockpit flight controls and displays that are not part of DAIS. The controls include the throttle, stick, rudder control and discretes. The displays are the basic flight instruments including the standby ADI, altimeter and warning lights. The CBIS interfaces to the ITB by means of the backup C/D interface unit which provides digital to analog conversions and a digital link to the SSDF (URT) PDP 11/40.

5.2.10 Four Port Buffer Memory

A four port buffer memory is used by the SSDF software for data transfer between the PDP 11 processors and the DEC-10 DMA window. The four port buffer memory consists of a unibus port interface, multiplexers, cyclic priority system and memory. The four port buffer memory unit appears ready for use by any port, with a worst case access time for any port user of 1 microsecond. The memory operations that may be performed are read word, write word, and write byte. There are no "hardware" protects built into this memory system, which means that if one user stores information in one location, that same location can still be accessed by another user. This memory may be assigned to any vacant 4K segment of the PDP 11 unibus address map.

5.2.11 HAS Super Control and Display Unit

The HAS Super Control and Display Unit (SCADU), in conjunction with the four port memory, is used to interrupt and pass control information (1 data word) between the PDP 11 processors. The HAS SCADU is used by the SSDF (URT) software to interrupt the SSDF (SSIU) software each minor cycle.

5.2.12 ITB Equipment Racks

The ITB equipment racks, as shown in Figure 29, include the ITB hardware as specified above as well as power supplies; special interface panels (multiplex data bus patch panel, Processor/BCIU breakout test patch panel, and CIU/RS-232 patch panel); power control panels and processor control panel.

5.3 ITB Support Software and PDP 11 Processors

Support software is resident in the PDP-11 processors.

5.3.1 Performance Monitor and Control

The Performance Monitor and Control (PMC) software resides in a GT44 System (PDP 11/40) under RT 11, and is used to set up, control and monitor the mission software resident in the DAIS processor. The PMC is capable of servicing from one to four DAIS processors at one time while simultaneously monitoring the multiplex bus traffic via the BMU and recording selected DAIS system data for post test analysis.

The PMC controls the SCADU's, CIU's, and BMU. The DAIS Processors are monitored and controlled through the SCADU's and the CIU's. The PMC starts and stops the set of DAIS Processors. The DAIS processor mission software is loaded from or dumped to the DEC-10 host simulation processor or the PDP 11 processors under PMC control.

The PMC sets up test cases which define the procedures and data collection for a simulation run. An operator is able to set up the test by means of an interactive interface and the creation of a test control file. The test control file is correlated with the DEC-10 simulation so that the collection of test data will proceed as defined by the file as the simulation progresses.

The PMC supports the system users in the debugging and evaluation of mission software by allowing selective real time and non-real time gathering of data from the DAIS processor and the multiplex data bus. The PMC provides a repertoire of commands to perform the non-real time and real time functions as follows:

1. Manipulate files on the DEC-10 and PDP 11 such as rename, delete, copy, etc.

2. Load or dump DAIS processor mission software from or to the PDP 11.

3. Perform the CIU functions as defined in paragraph 5.2.5 when the DAIS processors are halted.

4. Start and stop (halt) the DAIS processors and BCIUs.

5. Set up the SCADU to perform specified functions as defined in paragraph 5.2.4 during real time operation.

6. Set up the bus monitor to record or breakpoint, as defined in paragraph 5.2.3, on specific multiplex data bus messages.

7. Display, print, or log the data collected from the bus monitor, or DAIS processors via the CIU and SCADU. Data logged on a disk can be analyzed later by a Post Run Editor.

   The PMC provides the capability to examine or modify DAIS processor registers and memory locations either with absolute or symbolic addresses.

8. Provide interactive capability with the DEC-10 to set up and run the simulation model.

The PMC provides the capability to restart mission software, along with the simulation models, from a system snapshot point. This is accomplished by dumping mission software parameters at a specified PMC breakpoint, and reloading these parameters to restart the system. This snapshot and restart capability is used for repeated testing or demonstrations from a specific point in the mission profile.

5.3.2 Simulated Subsystem Data Formatter (URT) Software

The Simulated Subsystem Data Formatter (SSDF/URT) software receives and transmits data to and from mission software via the DAIS multiplex data bus to the real time simulation models running on the DEC-10 via the DEC-10 DMA window. The SSDF/URT sets up and controls the URT, double buffers the data in the four-port buffer memory, and provides a user selectable display of the multiplex data bus messages.

The SSDF/URT software is designed to run on both ITB and STS. The software will automatically check for the presence of CBIS and the SSDF/SSIU and control the interfaces to both. The SSDF/SSIU indicates it is on-line to the SSDF/URT by setting a flag in the four-port memory. This causes the SSDF/URT to generate two interrupts to the SSDF/SSIU each models computational cycle via the HAS SCADU. The first interrupt indicates the models output data is available in the four port memory and the second causes the SSDF/SSIU to swap the SSIU double buffers. The HAS SCADU master data register is used to distinguish between the two interrupts. The SSDF/URT also transfers the data received by the SSIU and remote terminals to the DEC-10 DMA window.

In non-real time mode, the SSDF/URT performs the following:

1. Memory management initialization.

2. Utilizes user defined tables to initialize the URT rams and registers for the specific mission.

3. Initializes the four-port buffer memory, HAS SCADU, display processor, CBIS and keyboard.

4. Accepts user configuration commands to specify the following:

   a. Whether or not the SSDF/URT software should keep cycling the real time models when mission software stops.

   b. Whether or not the major and minor cycle numbers should be passed to the PMC software via the HAS SCADU.

   c. Whether or not the URT should be set up to respond to minor cycle messages to bus address one (required for one processor configuration to respond to a dummy minor cycle message).

5. Simulate a mass memory device using the PDP 11 disk unit to allow loading of STS and ITB processors over the DAIS multiplex data bus.

6. Initialize a real-time clock to provide interrupts 32 times per second to drive the real time models.

The first minor cycle message received or a user command causes the SSDF/URT to enter real time mode. In real time mode, the SSDF/URT performs the following functions:

1. Maintains and displays minor and major cycle counts and passes them to the PMC software if required.

2. Generates interrupts to the DEC-10 to cycle the real time models based on timer interrupts.

3. Move data received from the URT and SSIU to the DMA window.

4. Controls the swapping of the URT double buffers and generates HAS SCADU interrupts to the SSIU to cause the SSIU buffer swapping.

5. Reads the models output data from the DMA to the four-port and generate HAS SCADU interrupts to the SSDF/SSIU to cause the data to be read into the SSDF/SSIU processor's main memory.

6. Uses the timer and DEC-10 interrupts to control access to the DMA window.

7. Inputs and outputs CBIS data (performing all necessary formatting).

8. Handles mode command interrupts from the URT as required to make the URT respond like a real RT.

9. Maintains displays of all pertinent information.

5.3.3 Simulated Subsystem Data Formatter (SSIU) Software

The SSDF/SSIU software is designed to run with the SSDF/URT software. The SSDF/SSIU sets up the SSIU to transmit and receive data from and to core memory. It moves the received data from core memory to the four-port memory and move transmit data from the four-port to core. The SSDF/URT will control the buffering of the data to/from the four-port from/to the DEC-10 DMA window.

In non-real time mode, the SSDF/SSIU performs the following:

1. Memory management initialization.

2. Utilizes user defined tables to initialize the SSIU rams and registers for the specific multiplex bus messages and RT activity.

3. Initializes the HAS SCADU, display processor and keyboard.

4. Accepts user configuration commands to specify which RT addresses are assigned to each SSIU slot.

5. Set a flag indicating SSIU on-line to the SSDF/URT software via the four-port memory.

The SSIU does not control the entry into real time mode but simply acts on SSIU and HAS SCADU interrupts. The software can be considered to be in real time mode when the first interrupt is received. In real time mode, the SSDF/SSIU performs the following:

1. The HAS SCADU "read" interrupt causes the models output data to be read into the PDP 11 main memory buffer not being used by the SSIU.

2. The HAS SCADU "swap" interrupt causes the SSIU buffers to be swapped.

3. SSIU interrupts (only generated on receive messages) cause the data received to be moved to the four-port memory (buffer A) and a HAS SCADU interrupt to be generated to the SSDF/URT with the four-port address of the data.

4. Activity words in the SSIU are set when asynchronous requests are received from the real time models.

5. Displays of pertinent system information will be maintained.

5.3.4 Evans and Sutherland Graphics System

The Evans and Sutherland graphics system is used to generate a cockpit out-the-window scene. The out-the-window scene is controlled by the simulation software so that the scene viewed from the cockpit has the correct dynamic orientation to synchronize with the simulated aircraft motion. The graphics interface software transforms aircraft attitude and position data obtained via the DEC-10 DMA window from the simulation software program into a form that can drive the Evans and Sutherland graphics system.

5.3.5 ITB PDP 11 Processors

The PDP 11 processors include:

1. Each PMC, SSDF (URT), and SSDF (SSIU) processor has a complex consisting of one DEC GT-44 PDP 11/40 processor, and other peripherals.

2. Evans and Sutherland Graphics interface processor consisting of a PDP 11 processor.

5.4 STS Support Hardware

The following hardware subsystems are identical to that described for the ITB:

1. One Universal Remote Terminal
2. One Bus Monitor Unit
3. Two Super Control and Display Units
4. Two Console Intelligence Units
5. Two Hazeltine Terminals
6. One Four-Port Buffer Memory
7. One HAS SCADU

The STS Power Distribution and Control (PDS) system controls, distributes and filters the power to each of these DAIS core elements and the STS support hardware.

The STS Test Control Center (TCC), shown in Figure 26, provides a centralized point for control of the entire STS. The TCC contains the terminals which can control the PMC and SSDF PDP 11 processors, and the DEC-10.

5.5 STS Support Software and PDP 11 Processor

The PMC software for STS is identical to the ITB PMC support software as described in paragraph 5.3.1.

The SSDF (URT) software for STS is identical to the ITB SSDF (URT) software except interfaces for the SSIU and CBIS are not operated.

The PMC PDP GT-44 System and SSDF (URT) PDP GT-44 System are similar to the ITB PDP 11 processor.

5.6 DEC-10 Host Simulation Processor

The DEC-10 facility block diagram is shown in Figure 31. The KI DEC-10 CPU is used with the interfaces and peripherals shown in the block diagram. The DEC-10 facility is a conventional DEC configuration except for the Direct Memory Interface Controller which provides a DMA window access to the DEC-10 memory for data transfer between the DEC-10 and the PDP 11 processors.

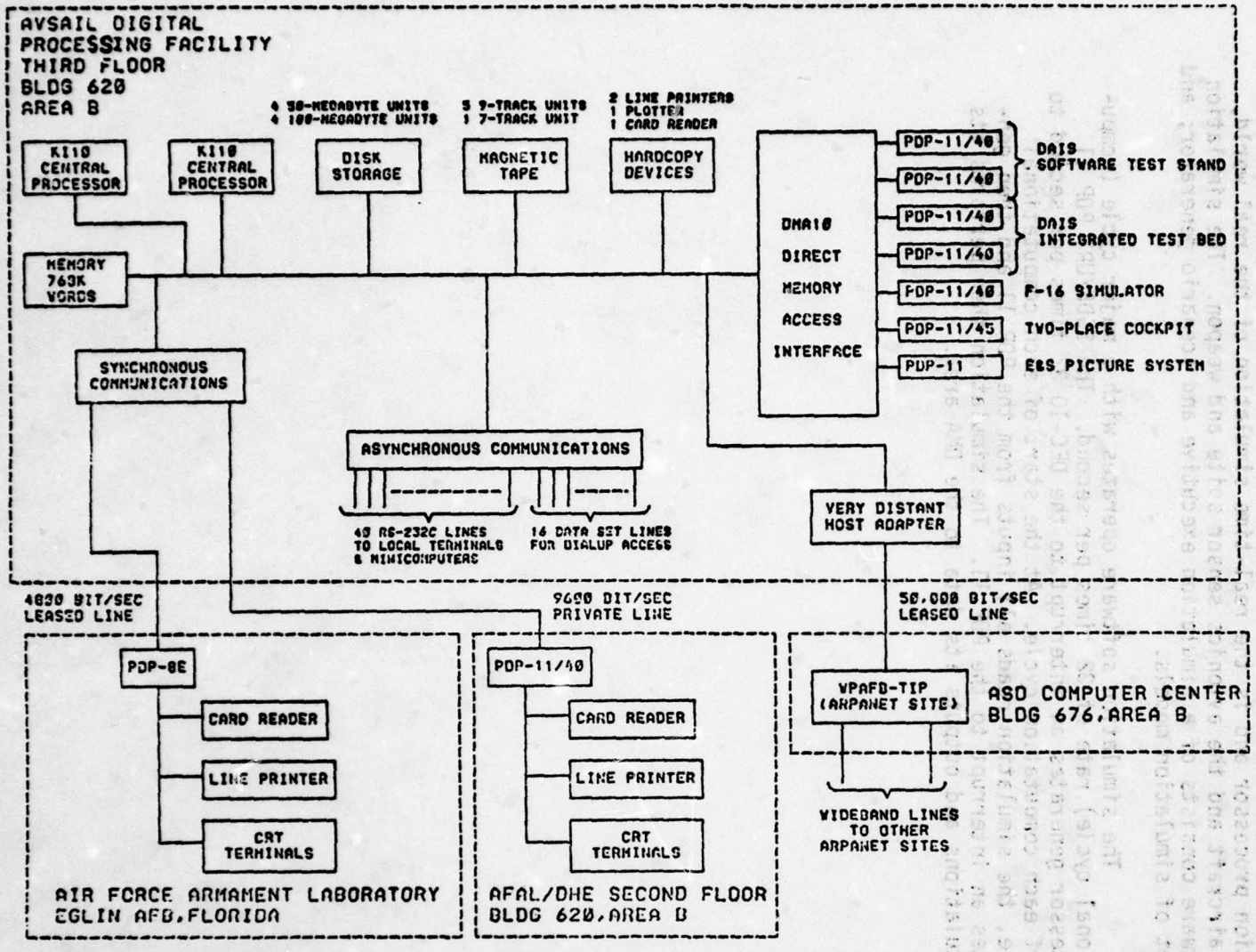


FIGURE 31. DEC-10 Host Simulation Processor Configuration

5.7 Simulation Software

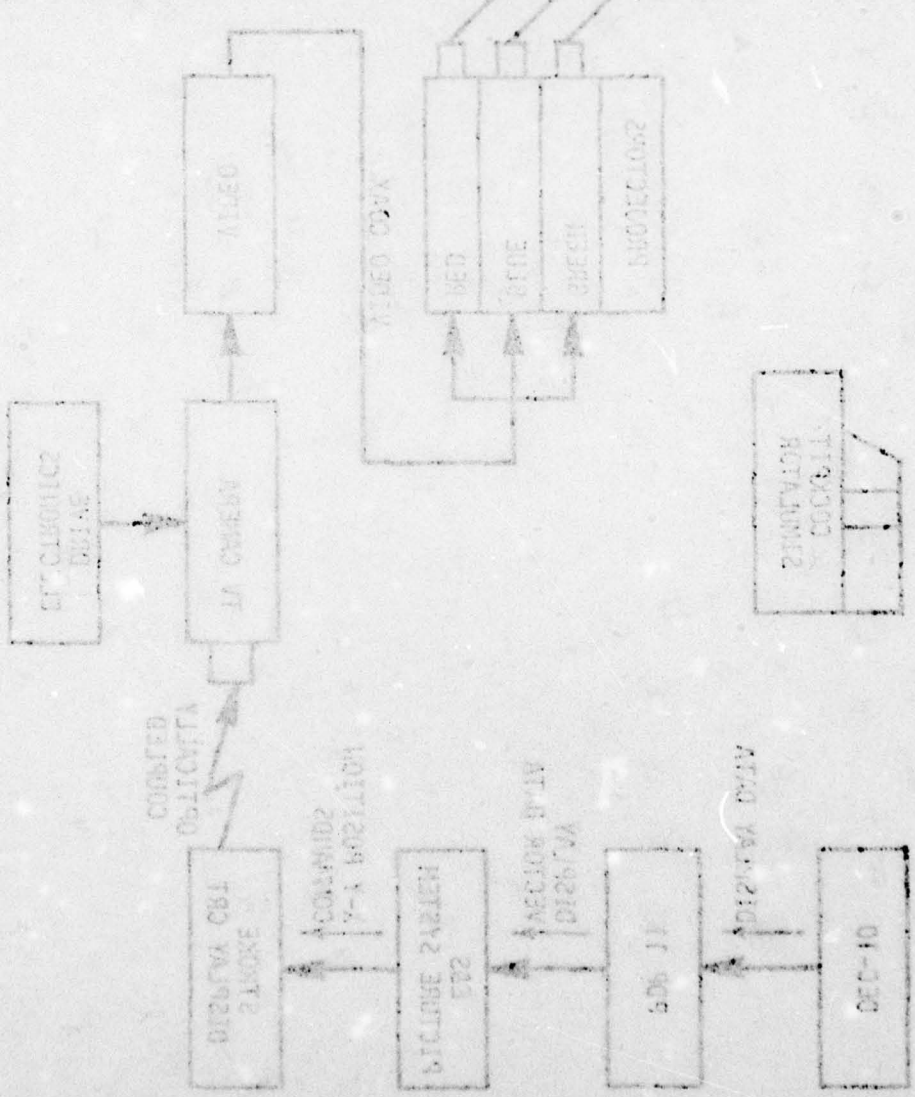
The Simulation Software is resident in the DEC-10 host simulation processor and is the real-time simulation of the real world, the aircraft and the avionics sensor suite and weapon. The simulation software consists of a simulation executive and scenario generator; and a set of simulation models.

The simulation software operates with a major cycle (computational cycle) rate of 32 times per second. The SSDF/URT PDP 11 processor generates an interrupt to the DEC-10 32 times per second to start each computation cycle. At the start of each computational cycle, the simulation reads all inputs from the PDP 11 and then generates an interrupt to the PDP 11. The simulation then performs its calculations and outputs its data to the DMA area.

5.8 Picture System

The picture system shown in Figure 32 consists of a simulation software controlled ground display graphics generator and projector system. The graphics generated display of the ground as seen by the "pilot" in the simulated cockpit, is projected on a motion picture screen in front of the cockpit.

An Evans and Sutherland graphics system generates an idealized pattern which represents ground terrain, targets, runway, etc. The Evans and Sutherland is controlled by the DEC-10 simulation software via the PDP 11 processor. The graphics are generated on a stroke CRT display and monitored by a closed circuit television camera. The TV image is transmitted to the Advent projection subsystem.



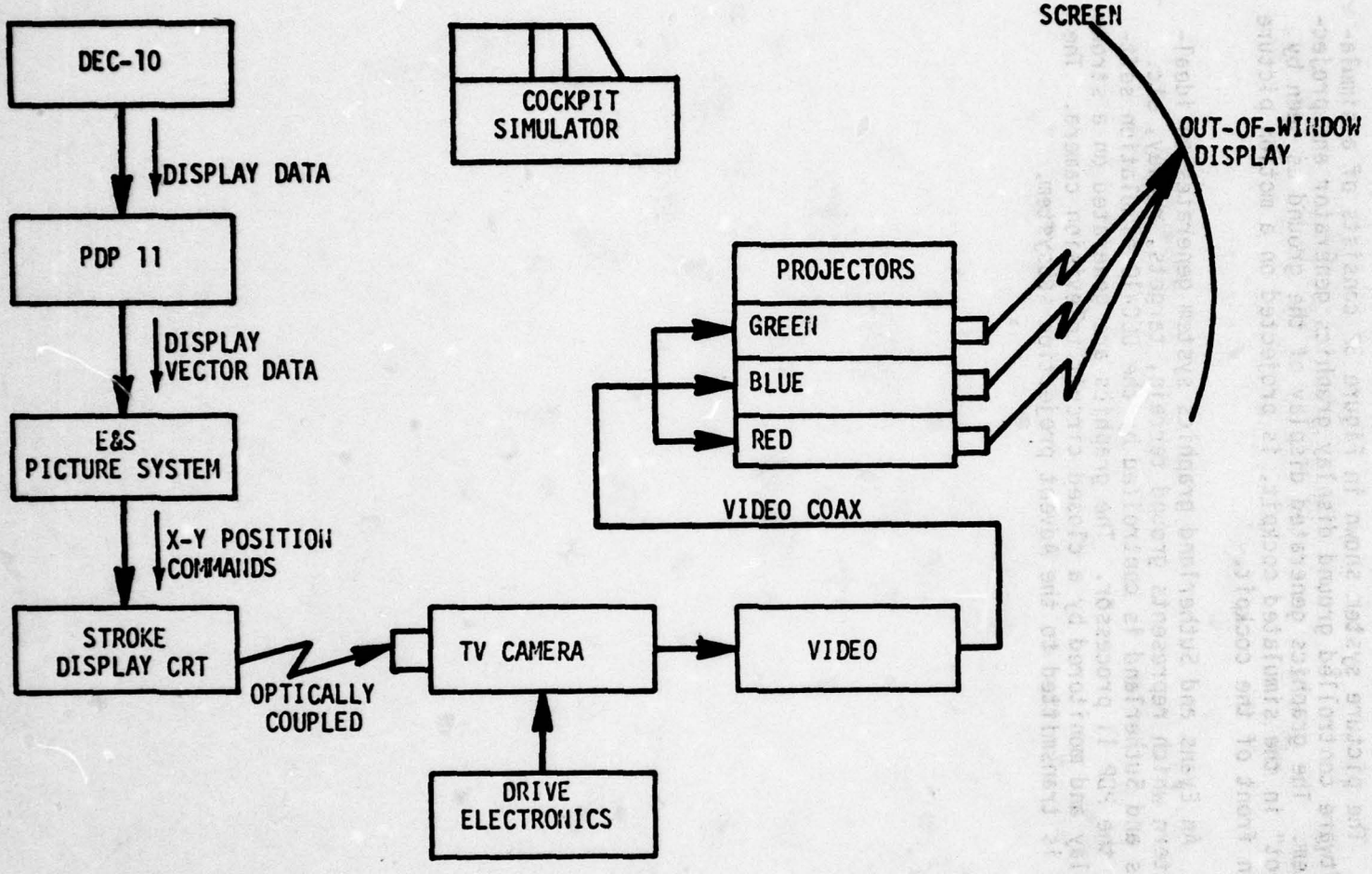


FIGURE 32. FUNCTIONAL CONFIGURATION - PICTURE SYSTEM

6.0 INTEGRATION AND TEST

The modular, well defined structure and interfaces of the hardware and software elements in the DAIS System Architecture greatly facilitates the integration and test. A step-by-step building block approach was taken to integrate and test these elements, for both the DAIS core elements, as well as the support facility hardware and software which supported the real time demonstrations, Figure 33. Initially, standalone tests were performed on individual elements, and then these elements were incrementally integrated and tested until the full system was completed for the initial demonstration. The following section highlights several of the key techniques which contributed to the successful and rapid integration and test activities.

6.1 Standalone Tests

6.1.1 Core Element Hardware

As each core element subsystem was received from the contractor, standalone acceptance tests were performed, first using the contractor supplied test support equipment and software. These tests were then augmented with AFAL and SITC tests. These tests were developed to verify all the electrical and functional interfaces of each subsystem. In several cases, the contractor supplied tests were not very comprehensive, and the test support equipment did not allow complete dynamic testing of all the functional interfaces. As a result, specifically for the multiplex equipment, complete acceptance testing of these core elements was not completed until the subsystem was integrated and tested with other system elements as discussed in the hardware integration and test section below.

6.1.2 Executive Software Testing

The DAIS Executive was also tested in an incremental fashion. A set of application tasks, called "stubs", were developed which would exercise the various Executive functions: pure executive service functions, synchronous bus control, asynchronous bus control, timing functions, and for both the remote and master operations. First, the local executive was tested using the SDVS statement level simulation on the DEC-10. Following that, the local executive was integrated with master executive and tested as a one processor (master) configuration, again incrementally with added "stubs" to test the various functions. This testing was performed on the processor/BCIU using support hardware as the "terminals" to receive or transmit bus messages. After the single processor tests were successfully completed, key "stubs" were repartitioned to the second processor to test the remote operation of the local executive, again incrementally, until all the functions were successfully tested. The next phase was partitioning of additional "stubs" to a third processor and repeating the set of tests to verify a three processor operation. Numerous problems were uncovered in the Executive, PALEFAC, processor, and BCIU. After these problems were corrected, all the features performed as stated in the Executive Specification as summarized below.

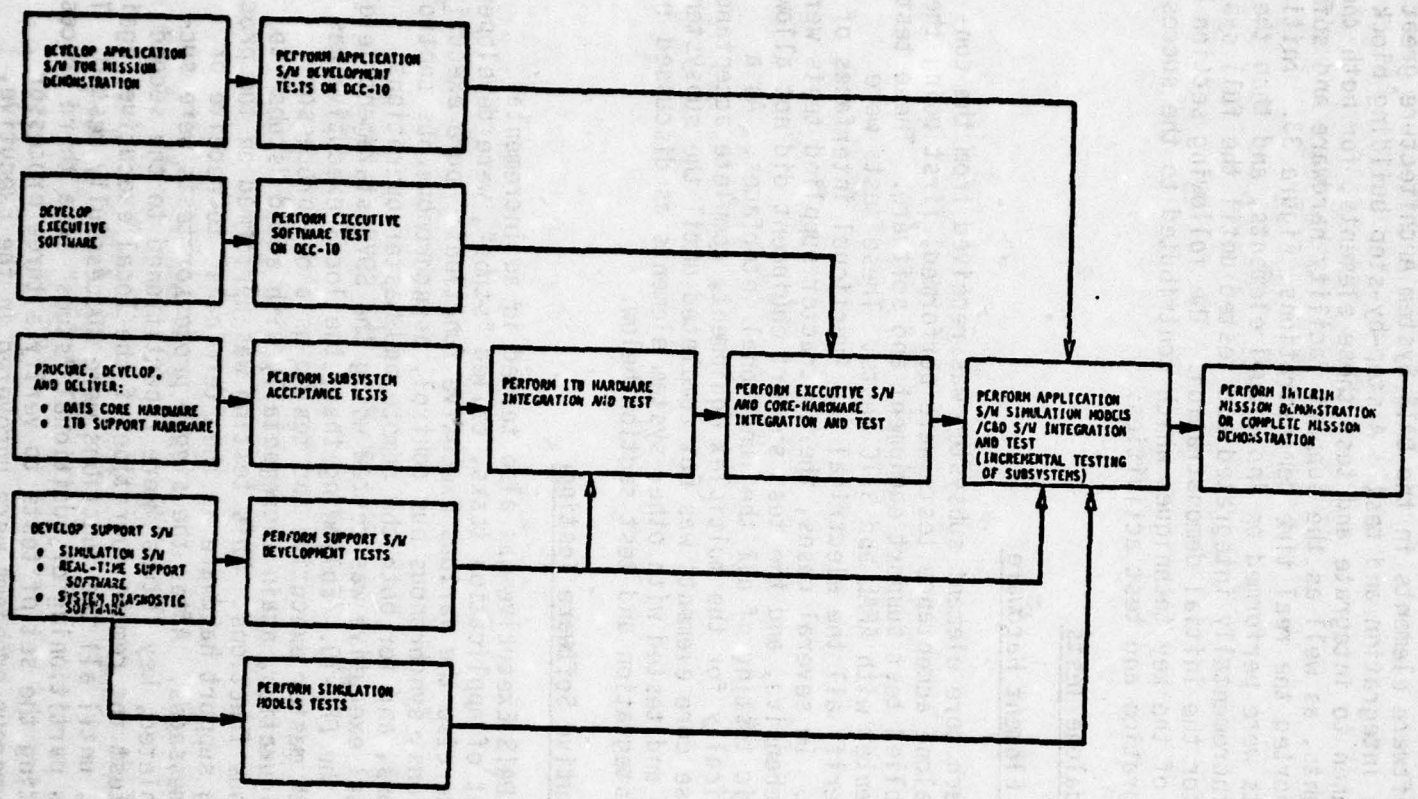


FIGURE 33. SYSTEM INTEGRATION & TEST STEPS

1. Synchronous bus traffic between master processor, remote processors, and the URT (simulated RT) operated as required.

2. Asynchronous bus traffic operated as required. This includes TRIGGERs from master and remote processors, asynchronous WRITEs, BROADCASTs with all the possible combinations of processors and RTs.

3. Event handling, both latched and unlatched, operated as required. This includes interprocessor and intraprocessor signals, waits, compool update events, dispatching, task priorities, and all their interrelationships.

4. Cancels and terminates operated as required for both interprocessor and intraprocessor requests.

5. Synchronous reads and writes operated as required. This includes resolving conflicts when a real or write request occured during the update minor cycle for that compool.

6. The real-time read, and wait on time performed as required in both the master and remote processors.

A limited amount of error testing was done in a separate test. This consisted entirely of forcing 'busy' conditions in the remote processors, then checking to be sure the error processing handled the condition properly and the data was received as required. All possible combinations of synchronous and asynchronous bus traffic, involving remote processors, were tested. The executive was able to handle the retries successfully in all cases.

6.1.3 Application Software Unit Test

Unit Test of the Application Software was performed on the Host Computer System. At this level of testing, the main concern was whether or not the logical operation of a single task is correct. A version of the DAIS Executive was hosted on the Host Computer System, and by compiling the task for the Host Computer Code Generator, running the task under test through PALEFAC, and then linking the task with the Executive, a test set was created. This set was executed on the Host Computer System. The user can specify inputs, cycle the task with the executive, and record the outputs. The user can also define lists of inputs, perform several cycles of execution, and record the list of outputs. As a result, once internal unit testing has been completed, an effective automatic or regressive testing capability

was developed for use on any version of that task, to reverify the task after changes were made. This entire testing process was performed at the functional level, since execution is not occurring on the processor system. This mode of execution was also applicable to groups of modules such that module interface compatibility tests were also performed on the Host Computer System.

6.1.4 Environmental Model Tests

The models execute on the Host Processor (DEC-10) and provide inputs and outputs to the multiplex data bus via the support hardware. Testing of the models was, therefore, limited to verifying the response, outputs of the individual models to input stimuli. This testing basically verified the interfaces and limited performance operation of the models set. Full dynamic testing of the models over a wide range of inputs is yet to be performed.

6.2 Hardware Integration and Test

Incremental integration and testing of each subsystem was performed, as they became available, until the system was fully integrated and tested. The key to the successful integration and test was the development of extensive test software. Two major test software programs were developed: System Diagnostics Software and Command Generation Software. The System Diagnostics was developed in a structured and modular fashion so additional tests could be readily added. Several of the key features of these software test programs were:

- Testing of all functional areas of the core elements under test.

- Testing of functional areas with extensive data patterns with error checking under test software control. Many failure modes were bit pattern sensitive and/or intermittent requiring large number of data patterns at the system operating speed in order to detect and isolate.

- Segmenting test software to functional areas of the hardware under test. This facilitated problem isolation by allowing the test operator the option of running test software on the failing segment only.

- Providing stop on error and loop on section, subsection, and failing data pattern. This facilitated problem isolation and provides conditions under which a problem (even intermittent), could be "scoped" or traced with a logic analyzer.

- Generating data patterns at execution rates at least comparable to the normal operation rate for the equipment under test. Many failure modes were induced only at or near maximum data rates because of noise, signal line "ringing" or "cross talk" and other high speed phenomena.

- Capability to sequence through all phases of the test repetitively (looping) with error checking. Capability allowed equipment testing for thermal, noise, and other intermittent errors, and provided a high degree of confidence when equipment operated successfully for extended periods of time.

- Support system verification prior to loading and running mission software, and supporting problem detection and isolation during mission software checkout and predemonstration phases. System integrity was established before loading mission software so mission software debugging would not be plagued with hardware problems.

- Structured and modular test software to facilitate development in stages and to allow inclusion of test software from various sources.

6.3 Application Software - Integration and Test

Once interface compatibility testing was performed upon a set of tasks that formed an application software subsystem (such as Navigation), the next step was to compile those same tasks for the DAIS processor, generate a load for the system, and test that load executing in real time interfacing with the sensor devices and Controls and Displays equipments over the multiplex bus. As each subsystem was checked out, the next subsystem was added to the test set as a unit and another set of tests for the added subsystem and the added sensors was run. The result was that system integration was completed when the last subsystem was integrated and tested. The integration process went extremely smoothly, and the major problems were in the areas of the interface definition with the simulated sensors, the interface definition with the Controls and Displays equipment, problems found in the Compiler Code generator for the DAIS processors, and with the instruction execution on the DAIS processor. None of these problems involved functional redesign of the Application Software, and thus illustrated the ease with which system integration and testing can be done with this software development approach.

7.0 REPRESENTATIVE APPLICATION

7.1 Mission Demonstration

The overall DAIS objectives are to (1) reduce unnecessary development proliferation, (2) improve operational efficiency including availability/maintainability, and (3) improve flexibility to changes. These objectives must be achieved without sacrificing the mission avionic functional capability attainable with conventionally configured complements of sensor and weapon subsystems. The satisfaction of these objectives and the requirement for acceptable system functional capability has been illustrated by successfully operating DAIS under simulated mission conditions. Specifically, the mission demonstration has illustrated: (1) the pilot interface with Controls and Displays (C&D), (2) the operation of the processors, mission software and multiplex system under realistic workload conditions, and (3) the ability of the system to perform weapon delivery and navigation functions.

The DAIS demonstrations were performed in the Integrated Test Bed facility. The purpose of this facility is to test DAIS mission software and core element hardware under real-time conditions. The facility provides a real time simulation of a military aircraft performing an operational mission. The simulation generates the interface signals so that the DAIS equipment and mission software is subjected to a data signal environment which is nearly identical to actual flight.

Simulation software was developed to provide real time simulation of military aircraft in an operational environment including the aircraft dynamics, the aircraft sensors and weapon targets. The simulation is driven from the cockpit by an operator acting as a "pilot". This simulated cockpit is equipped with the DAIS controls and displays so that the various modes of a mission may be "flown" by a "pilot" with an out-the-window background scene.

The Mission α demonstration was successfully performed during September 1978, to demonstrate the DAIS system. This demonstration is the first of the four scheduled Close Air Support (CAS) demonstrations. This initial demonstration included functional capabilities such as: Inertial/Baro-Damped Navigation; Command Navigation, TACAN and ILS Steering; MK82 Weapon Delivery (with weapon scoring); Stores Management; and preflight, takeoff/climb, cruise and approach/land checklists.

The simulation was flown from the DAIS cockpit using active DAIS Controls/Displays and simulation models which reside on the DEC-10, as shown in Figure 23. No real sensors were used in the demonstration. The avionic system consists of a two processor DAIS configuration as shown in Figure 34 and Table 11. All communication between the DAIS

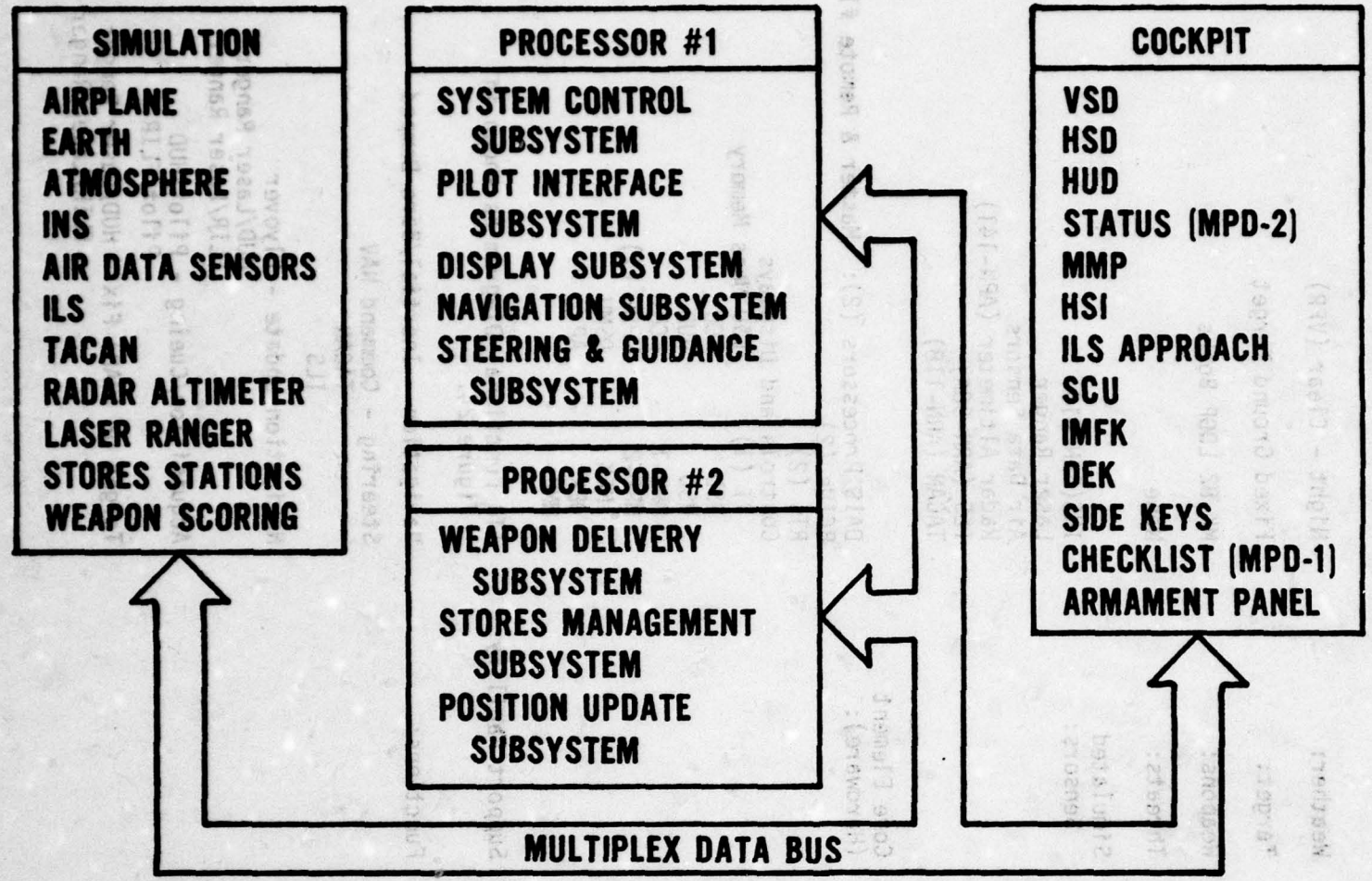


FIGURE 34. MISSION α CONFIGURATION

TABLE 11. MISSION α CONFIGURATION

Weather:	Night - Clear (VFR)
Target:	Fixed Ground Target
Weapons:	MK-82 LDGP Bombs
Threats:	None
Simulated Sensors:	INS(SKN2416) Laser Ranger Air Data Sensors Radar Altimeter (APN-141) ILS (ARN-58A) TACAN (ARN-118)
Core Element (Hardware):	DAIS Processors (2): Master & Remote #1 BCIUs (2) RTs (2) Controls and Displays RT (1) — C&D Mass Memory VSD — SCU HSD — HUD MPD-1 — PCP MPD-2 — MPDG (1) IMFK — DSMU DEK — AP MMP
Support Facility:	ITB Functional Diagram as shown in Figure 24.
Functions:	Navigation - Inertial/Baro-Damped Steering - Command NAV TACAN ILS Navigation Update - Flyover HUD/Laser Ranger FLIR/Laser Ranger Acquisition/Cueing - Pilot/HUD Pilot/FLIR Target or (OAP) Fix - HUD/Laser Ranger FLIR/Laser Ranger

TABLE 11. MISSION α CONFIGURATION (CON'T)

Weapon Delivery -	CCIP/Auto
	CCIP/Manual
Stores Management	
Communications -	UHF
Checklist	

processors/Bus Control Interface Unit, the DAIS cockpit and the simulation models was accomplished over a dual redundant MIL-STD-1553A multiplex bus using DAIS/1553A system protocol. A DAIS remote terminal was used to interface the DAIS cockpit electronics to the dual redundant 1553A multiplex bus. The subsequent Missions will employ additional sensors, such as Pave Penny and VATS/Pave Tack, to demonstrate additional navigation updates, target acquisition/queueing, and weapon delivery functions. Also, additional features to demonstrate system startup/restart, recovery, backup, and reconfiguration will be performed in these subsequent Missions.

7.2 Application Software Partitioning

The Application Software required to implement the Mission α requirements was initially broken into the following major areas: Navigation, Steering, Weapon Delivery, Stores Management, Display Control, Display Driving, System Control, and Pilot Interface as shown in Figure 34. The performance requirements for each area were defined, based upon the System Segment Specification for the Mission α configuration, and, then, the initial system design began. The data interfaces to and from the external devices were first defined as Compool Blocks. Figure 35 provides an example of the Navigation subsystem structure and data flow. Then, the initial definition of the individual tasks was begun. This task layout was performed by defining the individual functions required, defining under what conditions and in what order these functions should occur, and then grouping those functions which were logically related and occurred under the same conditions together as Tasks. Refer to Figures 36, 37, and 38 for examples of Navigation Subsystem. Once data layout was completed, the design and development of the individual tasks began, as did the design of the system control for each subsystem. The design involved defining the modes of operation of each subsystem, how the controlling task would control the subtasks in order to implement these modes, and how the subsystem would report current operating modes as well as unusual conditions as outputs.

As the design of the system evolved, it became apparent that the system consisted of a synchronous portion which must execute continuously (within a given mode of operation) and an asynchronous portion that executes upon the change of a mode (Navigation Controller for example), or upon the performance of some operation (passing over the next waypoint in the flight plan for example), or upon pilot action (performing a position update for example). From the beginning of the system design it was important to establish time of execution and size requirements for each task, as well as to keep a synchronous task execution and synchronous bus message transfer plan (Figure 38). The plan defines how each of the pieces of the synchronous portion of the system operates, as well as how it executes with respect to the other pieces. Once the definition of the tasks and the data interfaces were completed, the last major task of system design, the system partitioning, started. This step consists of determining which tasks will execute upon which processors in the system. There are many factors which go into making this decision. These factors are: desire to keep subsystems together as a unit due to their generally high transfer of data between their individual tasks, desire to keep system control functions in the master processor to facilitate its communications with all of the subsystems, desire to keep pilot interface in the master processor to facilitate its communication with the pilot keyboard, and the desire to balance high CPU throughput tasks against high memory size tasks in order to avoid having one processor CPU time bound, and another memory

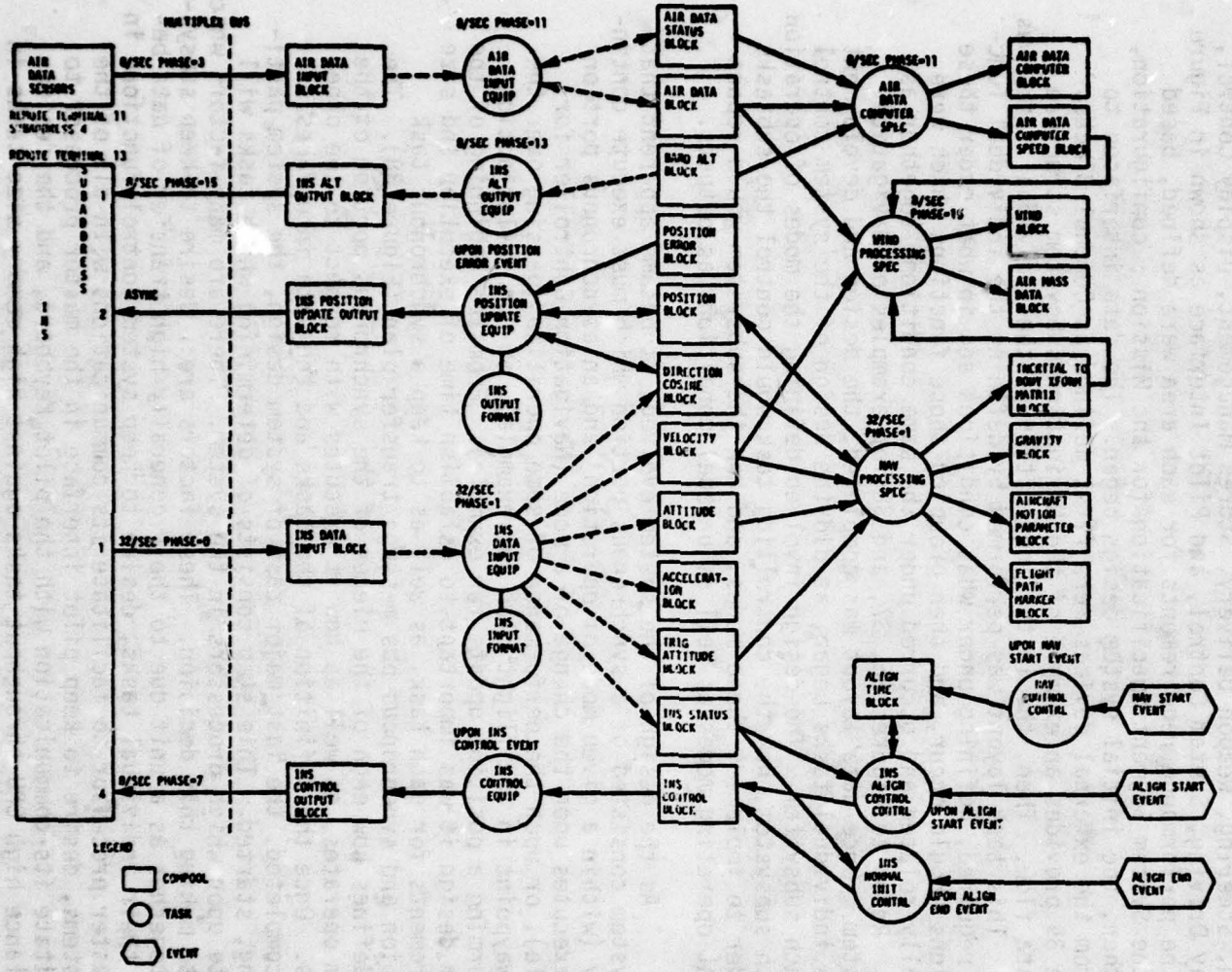


FIGURE 35. NAVIGATION SUBSYSTEM STRUCTURE AND DATA FLOW

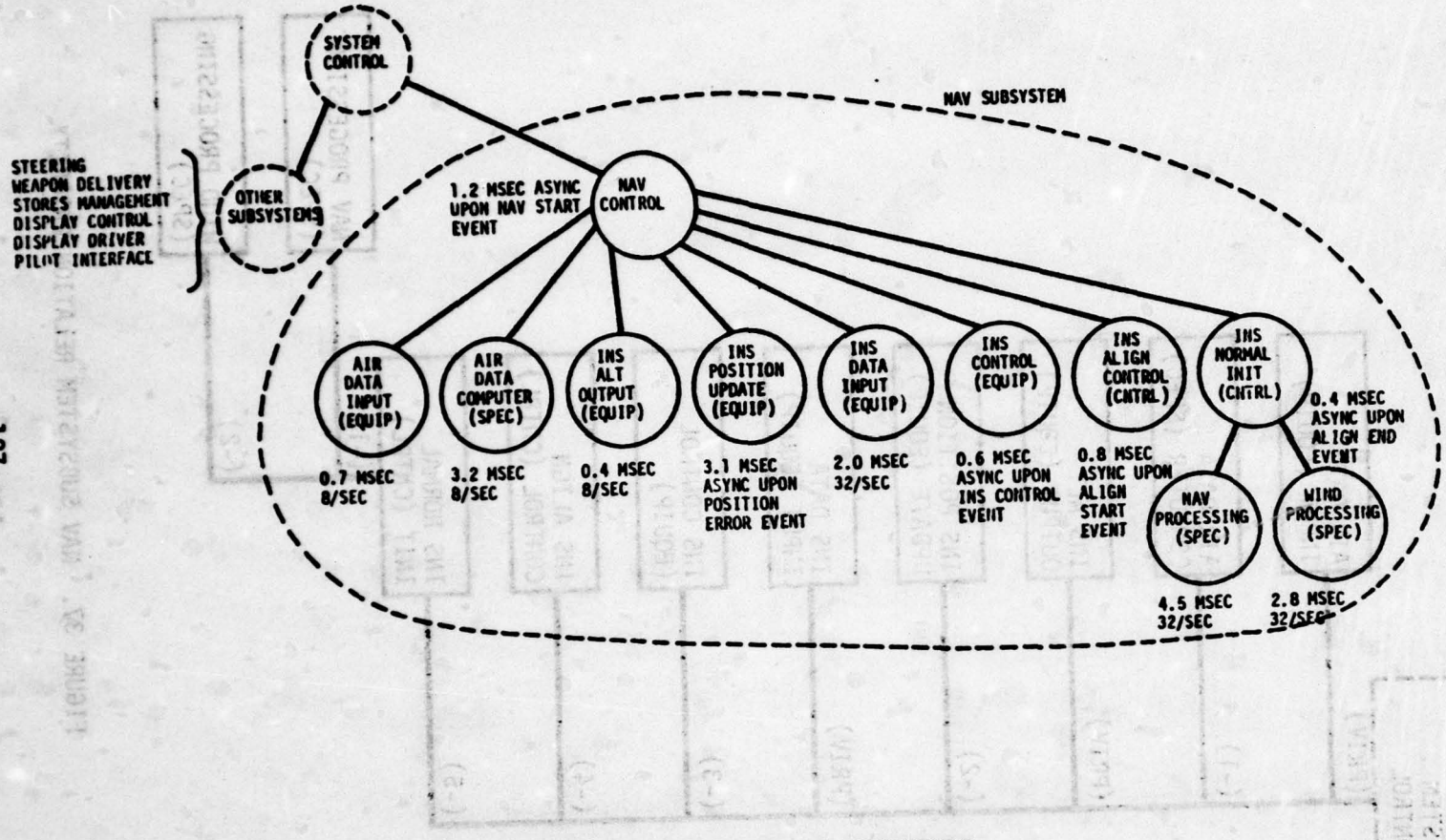


FIGURE 36. NAVIGATION SUBSYSTEM ARCHITECTURE

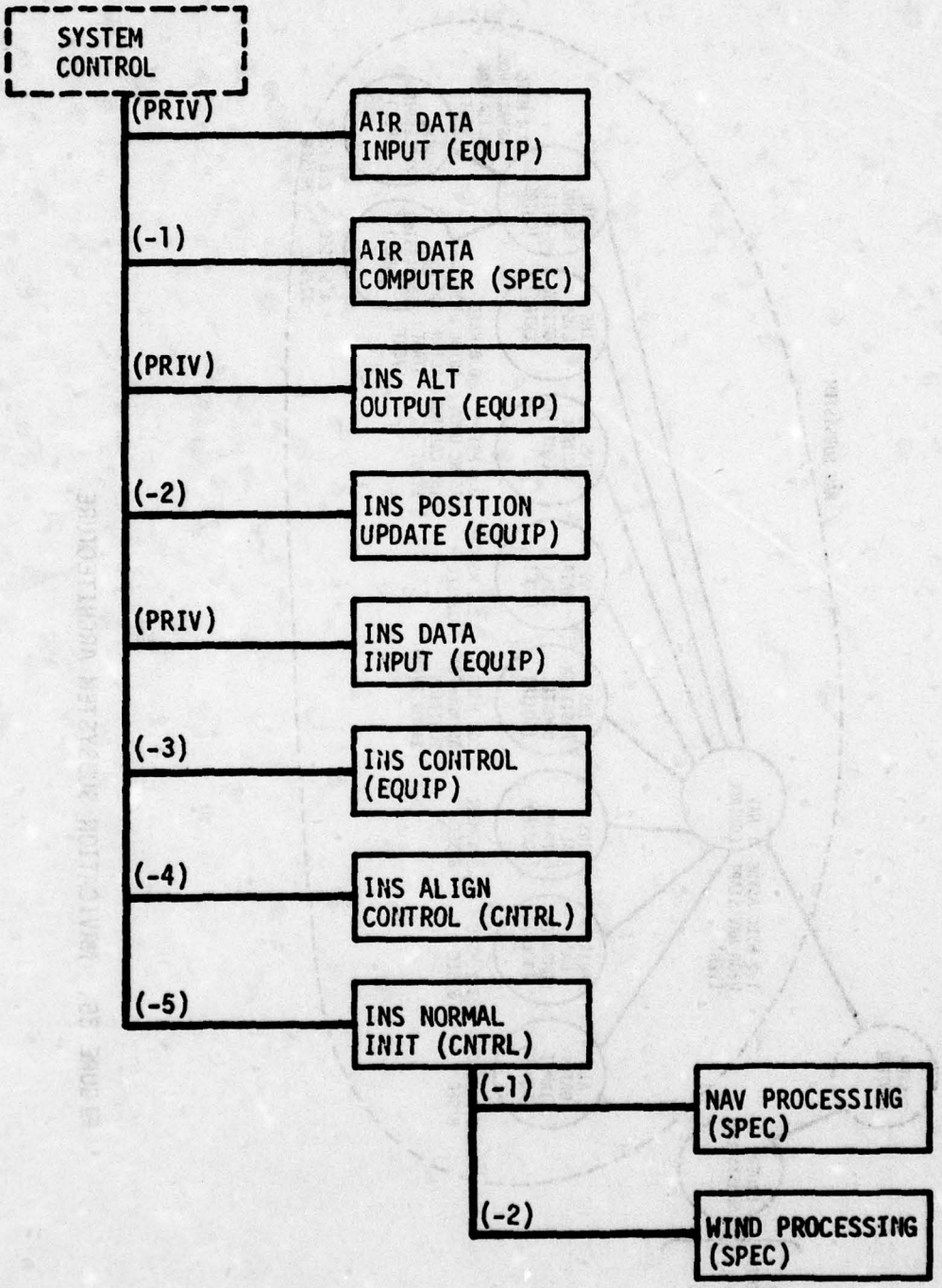


FIGURE 37. NAV SUBSYSTEM RELATION PRIORITY

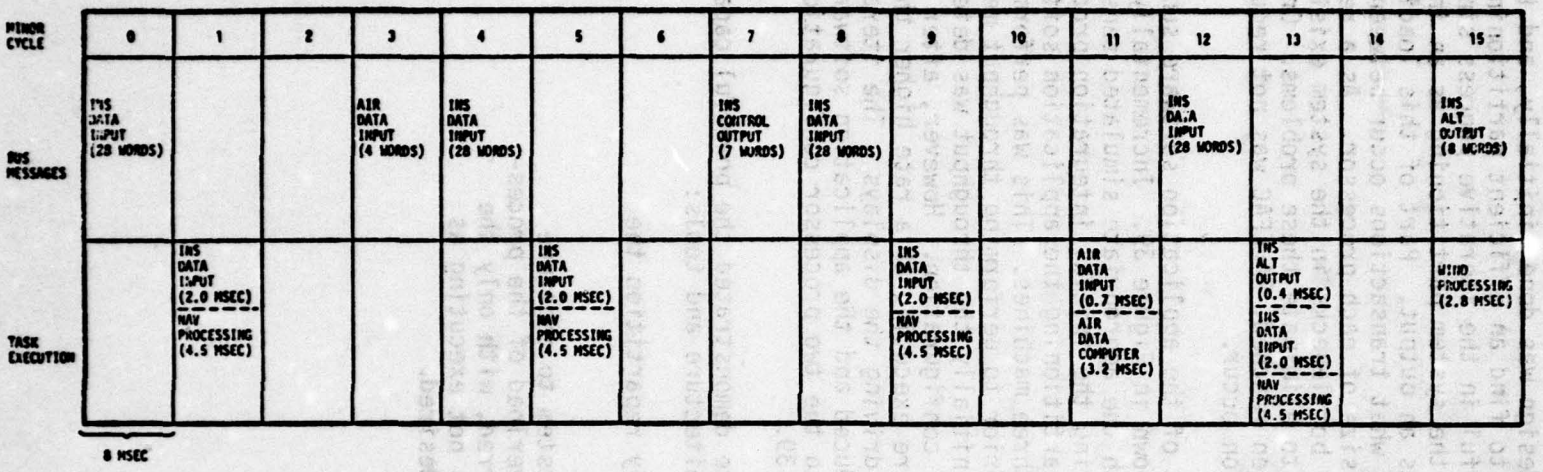


FIGURE 38. NAV SUBSYSTEM SYNCHRONOUS OPERATION (REPEATS EVERY 16 MINOR CYCLES)

space bound. This system layout design was done, initially, and then, repeated over many times in order to find an efficient partitioning. The PALEFAC tool was extremely useful in the iterative process since it accepts the user's description of the system partitioning as an input, and generates the system loading as an output. Part of this loading information was the declaration of what transactions occur between processors, as well as the memory size of each processor. As a result, the user begins to determine where bottlenecks in the system exist, and make system configuration changes to eliminate these problems. Once a stable system configuration has been reached, PALEFAC was not required unless changes in that configuration occur.

The integration and test of the application software subsystem was performed in 9 steps as shown in Figure 39. Incrementally, a subsystem was added and tested with the appropriate simulated sensor or control and display elements. During the initial integration process, three DAIS processors were used, partitioning the application software as shown in Figure 39 across all three machines. This was performed so the subsystems could be verified prior to performing throughput measurements on each application task. Initially the throughput was determined to be too high for a two processor configuration. However, after further examination, several subsystems were executing at a rate higher than required, specifically subsystems driving the displays. The iteration rates of these subsystems were reduced and the application software subsystems were all integrated into the two processor configuration, as shown in the final step in Figure 39.

This integration sequence demonstrated the powerful capabilities inherent in the DAIS architecture and tools:

1. Capability to readily repartition the system.

2. Capability of the system to continue operation even if overload of the processor throughput occurred, with only the lower priority tasks not executing as often as initially desired.

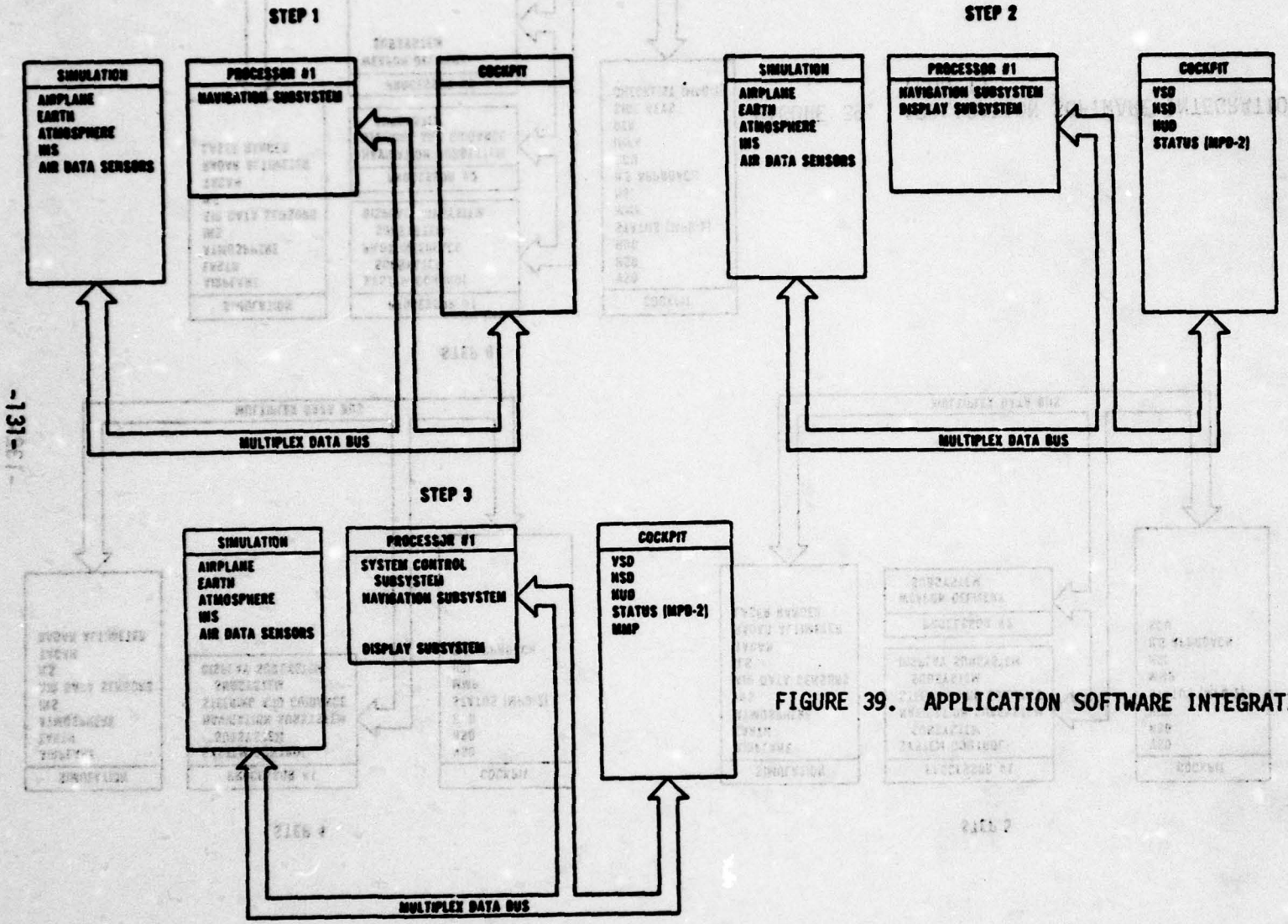


FIGURE 39. APPLICATION SOFTWARE INTEGRATION

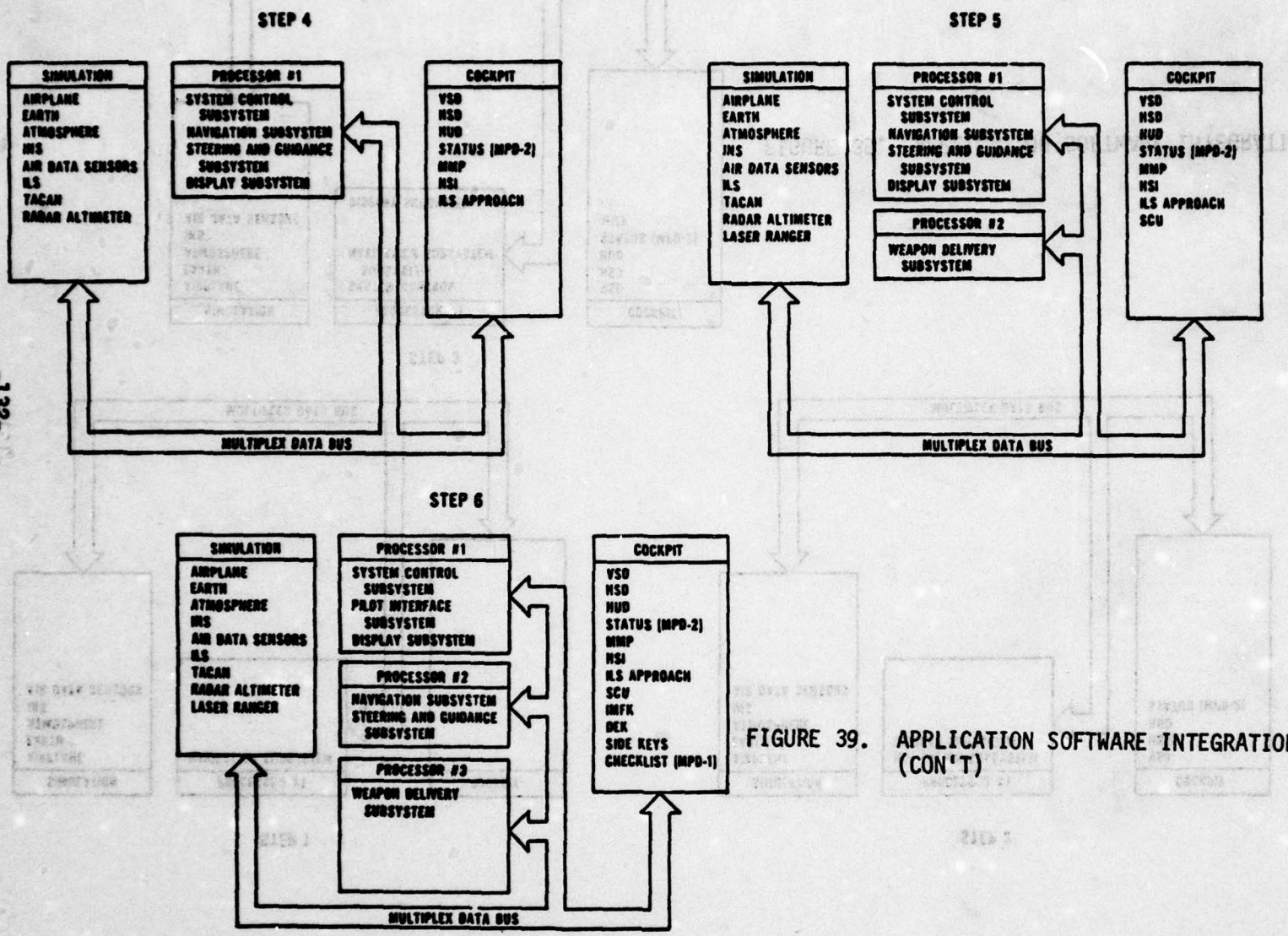


FIGURE 39. APPLICATION SOFTWARE INTEGRATION (CON'T)

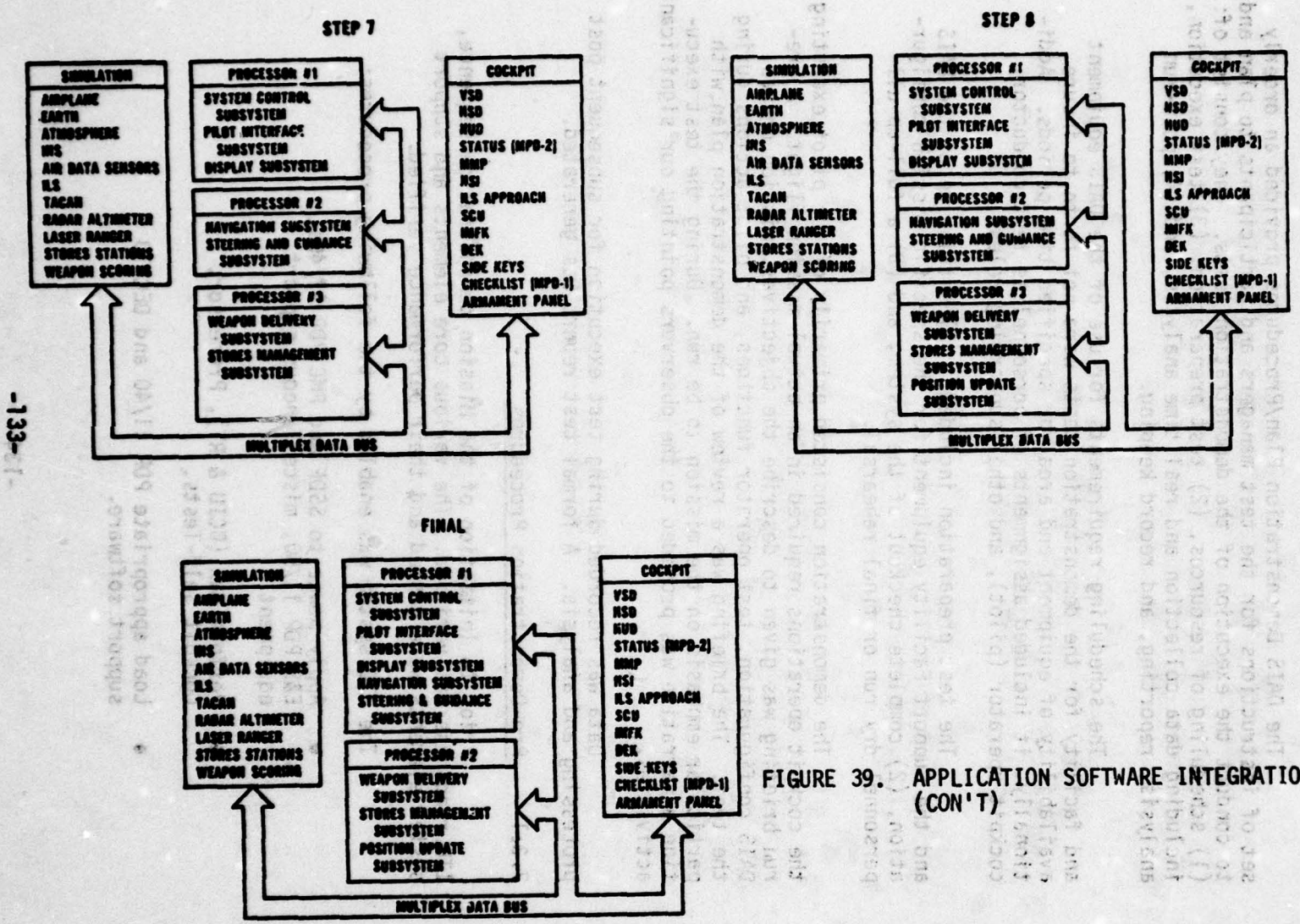


FIGURE 39. APPLICATION SOFTWARE INTEGRATION (CON'T)

7.3 Demonstration Procedures

The DAIS Demonstration Plan/Procedures provided an orderly set of instructions for the test managers and participants to plan and to conduct the execution of the demonstration tests. They consist of: (1) scheduling of resources, (2) test preparation, (3) test execution, including data collection and real time analysis, and (4) post-run analysis, reporting, and record keeping.

The scheduling requirements for use of the DAIS equipment and facility for the demonstration tests were set forth to insure availability of equipment and area for specified time periods. Additionally, it included assignments of personnel as test conductor, cockpit operator (pilot), and other supporting roles.

The test preparation included: (1) setting up of the DAIS and the Support Facility equipment for the specific mission configuration, (2) complete checkout of the system, (3) a full-up all-personnel dry run or final rehearsal.

The demonstration consisted primarily of the pilot executing the cockpit operations required in an actual mission flight. A pre-run briefing was given to describe the objectives, mission phases, DAIS configuration, test operator functions, and pilot actions during the test. The briefing was a review of the demonstration plan with particular emphasis on the mission to be run. During the test execution a narrative was provided to the observers pointing our significant activities.

Data was recorded during test execution for subsequent post processing and analysis. A formal test report was generated.

7.3.1 Pre-Demonstration Procedures

Prior to initiation of the Mission demonstration sequence, ITB readiness was verified. The various core elements and support system elements were tested and their performance verified.

The ITB system was enabled by the following procedures:

- Apply power to SSDF and PMC PDP 11/40s, E&S PDP 11/50, miscellaneous support equipment
- Perform Mux (BCIU & RT), Processor, Cockpit Self-Tests.
- Load appropriate PDP 11/40 and DEC-10 support software.

- Select and initialize Simulation Models
- Initialize SSDF, PMC, E&S processors
- Load Mission Software in DAIS Processor

After all support equipment was activated, the avionics processors and multiplex system were activated. The normal pilot sequence for these operations is:

- DAIS Bus Power Switches ON
- Processor Power Switches ON
- Check Processor Power Lamps Lit Green
- Check IMFK

After the system was determined to be operable, the pilot was notified on the IMFK: "System Ready".

7.3.2 Mission α Demonstration Procedures

The pilot sequences through the mission operations, using the integrated DAIS Controls and Displays (C&D) to display information from mission software about the mission and avionics functions, and to control operations to be performed by the software. Master modes set the display devices in certain nominal formats, which can be overridden by the pilot to cause alternate modes or operations.

The demonstration scenario for the mission is outlined in Table 12 and shown in Figure 40. Discussion of the various flight phases and operations is presented below.

7.3.2.1 Preflight

The system is initialized to a NULL master mode after all AN/AYK-15 processors and MPDGs are loaded, waiting for pilot selection of a master mode. The pilot is then prompted on the IMFK to select the Preflight Master Mode (PREFLT).

This master mode can be selected after initial program load until the weight is off the gear, provided no master mode other than Takeoff/Climb has been selected. (After weight is off the gear, the switch output is ignored by mission software until weight is on the gear). PREFLT CHKLIST on the IMFK can be selected using the Library specialist function without selection of the Preflight on the master mode panel. The checklist permits verification and/or modification of mission data and verification of take-off readiness.

TABLE 12. MISSION SCENARIO

MILESTONES	SEGMENT LENGTH (n.m.)	ALT. (K ft.)	AIR SPEED (kts.)	TRUE HEADING (deg.)	MASTER MODE	ACTIVITY
					PREFLT	• Verify Mission data and takeoff readiness
DAIS AFB ($34^{o}44'N$, $120^{o}34'W$)	15.3	1	350	19.9	T.O./ CLIMB	• Climb to cruise altitude • TACAN Steering • Flyover Update
TACAN Station ($34^{o}58'N$, $120^{o}28'W$)	14.9	1 to 5.2	350	134.9	CRUISE	• Command Nav Steering
Mountain Peak ($34^{o}48'N$, $120^{o}15'W$)	5.0	5.2-2	300-400	136.1	CCIP/Auto	• CCIP/Auto, Mk82 Delivery
Factory ($34^{o}44'N$, $120^{o}11'W$)	8.3	2-12	300-460	180.7	CCIP Man	• CCIP/Man, Mk 82 Delivery
Tank ($34^{o}36'N$, $120^{o}10.5'W$)	17.4	12-6	350	249.7	CRUISE	• Command Nav Steering • Laser Ranger Update
DAM ($34^{o}30'N$, $120^{o}30'W$)	14.3	6-0	350-160	348.0	APPR/LAND or PREC/APPR	• TACAN/ILS Approach & Land (Runway 36)
DAIS AFB						

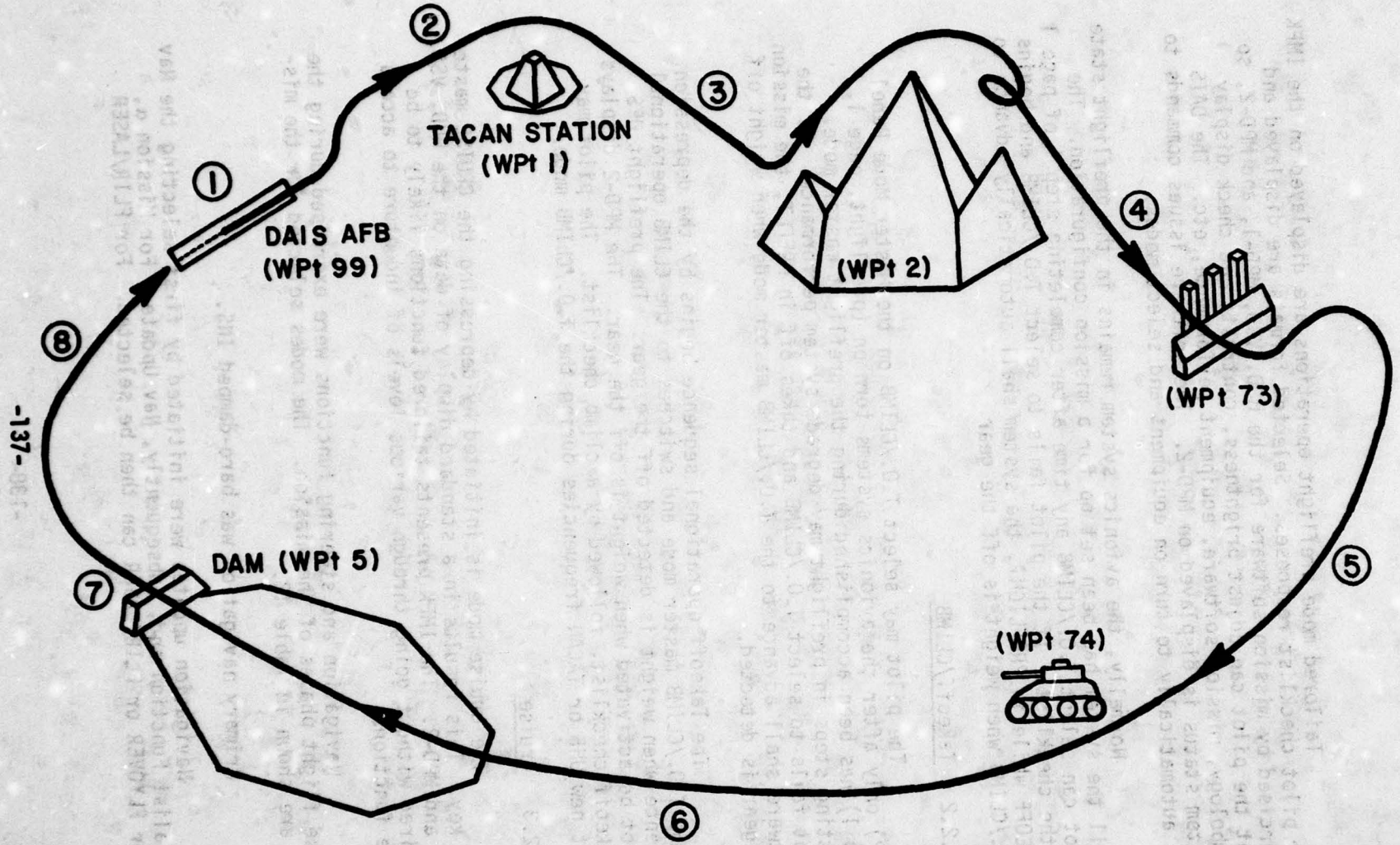


FIGURE 40. DEMONSTRATION SCENARIO

Tailored mode preflight operations are displayed on the IMFK for pilot checklist responses. Selected formats are displayed and exercised by mission software for the HSD, VSD, MPD-1, and MPD-2, so that the pilot can adjust brightness, contrast; and check display symbology, mission software, equipment performance, etc. The DAIS system status is displayed on MPD-2. The software issues commands to RTs automatically to turn on equipment and select modes.

Normally, the avionics system remains in the preflight state until the system has been set up for a mission configuration. The pilot can select T.O./CLIMB any time after completing step 1 of page 1 of the checklist. If the pilot fails to select T.O./CLIMB and begins TAKEOFF while in PREFLIGHT, the system shall automatically advance to T.O./CLIMB when weight is off the gear.

7.3.2.2 Takeoff/CLIMB

The pilot may select T.O./CLIMB on the Master Mode Panel (MMP) only after the avionics systems turn on (preflight, page 1, step 1) has been accomplished during the preflight master mode. Omitting steps in preflight may degrade system performance. If the pilot fails to select T.O./CLIMB and takes off in PREFLT, the mission software shall advance to the T.O./CLIMB master mode when weight off the gear is detected.

The Takeoff operational sequence begins by the depression of the T.O./CLIMB master mode and switches to the CLIMB operational sequence when weight is detected off the gear. The preflight OPS cannot be activated when weight is off the gear. The MPD-2 displays a takeoff checklist, followed by a climb checklist. The pilot may input new UHF or TACAN frequencies during the T.O./CLIMB mode.

7.3.2.3 Cruise

The cruise mode is initiated by depressing the CRUISE master mode key. This results in a standard display of data on the HUD, VSD, HSD, and MPD-2. The IMFK presents tailored functions likely to be required without going through various levels of indenture to access these functions.

Navigation and steering functions were exercised during the cruise flight phases of the mission. The modes selected for the mission are shown in Table 12.

Primary navigation was baro-damped INS.

Navigation updates were initiated by first selecting the Nav Specialist Function and, subsequently, Nav Update. For Mission α, either FLYOVER or FLIR/LASER can then be selected. For FLIR/LASER

updating, the Aiming Reticle (AR) is positioned over the update point on the HUD/FLIR display.* In both cases, the Sensor Control Unit (SCU) is used to designate the update point. Selection of UPDATE causes longitude and latitude to be updated to values computed by the selected update mode.

The default steering mode for the mission was Command Nav, which generates steering commands along a great circle route from present position to the next route point in the flight plan. The pilot may select a new waypoint from those in the flight plan by activating the FLY TO command on the CRUISE IMFK display and entering a new waypoint identification number. New waypoints can be introduced by selecting the Nav Specialist Function on the IMFK and by selecting Nav Data Entry on the subsequent IMFK display.

TACAN steering was selected in all missions by the TCN key in the Nav Specialist Function display. The IMFK responds with the NAV TACAN sensor mode control display. A channel change can be initiated by selection of TCN CHNG.

7.3.2.4 Weapon Delivery

Several weapon delivery techniques were exercised in the demonstration mission. Table 11 lists the capabilities available for the mission. Table 13 notes the weapon load options exercised in the mission. The weapon delivery options are discussed below.

a. Continuously Computed Impact Point/Automatic Release (CCIP/AUTO)

In this mode, the weapon delivery computations are based upon relative target data from a currently ranging sensor subsystem. Steering data is computed and displayed to guide the pilot to a release condition. Release is automatic with pilot concurrence. In the event of loss of track during this mode, the system will use alternate methods to compute release conditions. In this mode, an offset aim point may be used with pilot input of the offset range and bearing to the target from the aim point.

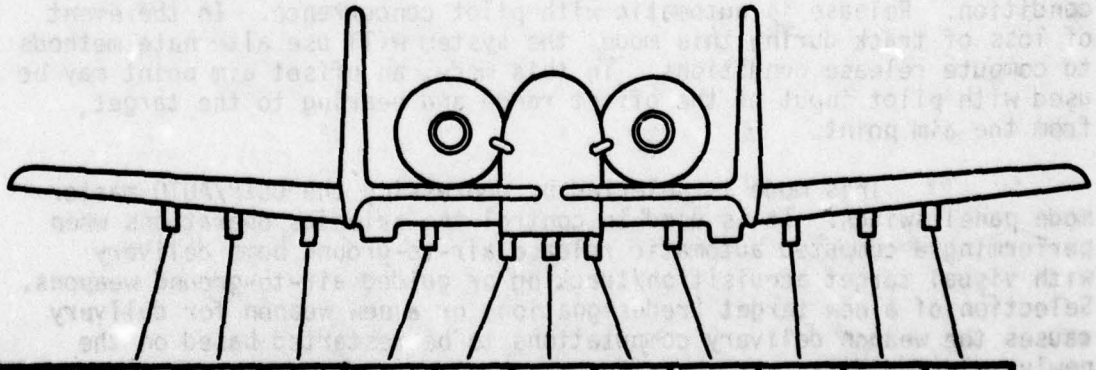
This mode is selected by depressing the CCIP/AUTO master mode panel switch. It is used to control the avionics operations when performing a computed automatic release air-to-ground bomb delivery with visual target acquisition/tracking or guided air-to-ground weapons. Selection of a new target (redesignation) or a new weapon for delivery causes the weapon delivery computations to be restarted based on the newly selected data.

*The HUD/FLIR display will be represented by the E&S display.

TABLE 13
WEAPONS LOAD OPTIONS

OPTIONS	QUANTITY	DELIVERY	INTERVAL	STATIONS
1	12	Singles	50	57
2	4	Pairs	100	39
3	6	Salvo	Default	8
4	3	Singles	1000	210

WEAPONS LOAD BY STATION



STATION	1	2	3	4	5	6	7	8	9	10	11
QUANTITY	--	3	3	6	6	--	6	6	3	3	--

Visual offset bombing is performed using this master mode and insertion of offset aiming point (OAP) data.

Selection of a weapon program provides a "quick change" page on the IMFK for the weapon selected. This page permits the pilot to change release parameters and then return to the tailored page.

After selecting a weapon delivery master mode, the bomb fall line (BFL) appears on the HUD. The BFL will pass through the flight path marker (FPM). The aiming reticle (AR) will appear stowed within the FPM, and the pull-up anticipation cue will appear 3.5^{o} below the FPM on the BFL. CCIP/AUTO can be selected to bomb any visually acquired target. However, if the barometric ranging mode comes into priority, the altitude and mean sea level pressure (MSLP) stored or a waypoint, which could have been selected under the FLY TO function during the cruise master mode, are used for slant range computations. If these values are not correct for the target being attacked, the desired accuracy will not be obtained.

When this master mode is selected, the HUD switch lamp is lit on the sensor control unit (SCU) which indicates the SCU controls the position of the HUD aiming reticle. The SCU can be used to move the aiming reticle in any direction. The BFL moves with the aiming reticle until designation. If the slewing is terminated, without designation, the aiming reticle becomes ground stabilized just as though target designation had been accomplished. Once the pilot designates the target with the designation button on the SCU, the aiming reticle is stabilized to remain on the designated point on the ground. Upon designation, the flight director is removed and the BFL will no longer intersect the aiming reticle, but will be halfway between the aiming reticle and the FPM, indicating the required lateral steering. As the aircraft advances toward target, the aiming reticle (and the target) may disappear from the pilot's field-of-view. Depression of the designate button a second time will undesignate the target and restow the aiming reticle and BFL on the FPM. Once the target is within range, the solution cues appear on the HUD, and weapon delivery may proceed.

If delivery of a weapon selected in this master mode is not completed and transition to another weapon delivery master mode occurs in which this weapon is a valid choice, the option remains active. If a non-weapon delivery master mode is selected, all weapons are automatically dearmed. The weapon options used in the mission are shown in Table 13.

Command steering from the previous master mode reamins active on the VSD until target designation, at which time attack steering is engaged. Attack steering is target referenced for both the VSD and HUD and steers to a release point.

b. Continuously Computed Impact Point/Manual Release (CCIP/Manual)

In this mode, the AR on the HUD will be located at the current bomb or gunnery impact point. The pilot then flies the aircraft to bring the AR into coincidence with the target. There will be two submodes to the CCIP/manual mode depending upon pilot selection of either bombs.

This mode is selected by depressing the CCIP/Manual master mode panel switch. It is primarily to control the avionics operations when performing an air-to-ground gunnery or rocket attack with visual target acquisition/tracking and manual release. It is also used as a backup mode to release gravity type weapons. Selection of a new target or a new weapon for delivery causes the weapon delivery computations to be restarted based on the newly selected data.

If delivery of a weapon selected in this master mode is not completed and transition to another weapon delivery master mode occurs in which this weapon is a valid choice, the option remains active. If a non-weapon delivery master mode is selected, all weapons are automatically dearmed.

Steering from the previous master mode remains active until target designation, at which time, attack steering is engaged. Attack steering is target referenced and steers to a release point.

7.3.2.5 Approach and Land

Approach and Land can be either non-precision or precision, the latter using ILS. The corresponding master modes are described below.

a. Approach and Land Master Mode

This master mode is entered by pilot selection of the APPR/LAND switch. It is used to control the flight operations for non-precision approach and landing. MPD-1 will display the selected approach plate for the airport and runway. MPD-2 will display the tailored status; the HSD wil display HSI symbology in TACAN steering or waypoint map in Command Nav Steering; the HUD and VSD will display the approach and landing symbology; and the IMFK will display the checklist and tailored specialist functions. After the pilot selects APPR/LAND on the master mode panel, the descent checklist appears on the IMFK. Upon completion, the IMFK displays the tailored approach and landing functions. The pilot advances to the before landing checklist if weight is off the gear by pressing the CKLIST key on the IMFK. After completing that checklist, the IMFK again displays the tailored functions. After touchdown (weight on gear) the pilot may select display of the post landing checklist. After completion, the IMFK displays the de-arming area and shutdown checklist.

b. Precision Approach

This master mode is selected by the pilot depressing the PREC/APPR switch. It is used when flying a precision approach using localizer and glide slope. Displays and information are: MPD-1, the selected approach plate for the airport and runway; MPD-2, tailored status display; HSD, the precision approach chart; HUD and VSD, the precision approach and landing symbology; and IMFK, the tailored specialist functions. Approach and landing checklists are callable through a tailored mode IMFK key as in APPR/LAND. The postflight checklist appears on the IMFK when selected by the pilot and weight is on the gear.

8.0 DAIS SYSTEM PROTOTYPE

8.1 Introduction

The purpose of this section is to describe the Digital Avionics Information System (DAIS) Prototype which was a simulation system used to investigate the baseline DAIS design.

The DAIS prototype project within the DAIS program was undertaken in order to demonstrate the baseline DAIS design including the design of its hardware and software elements. The particular objectives of the DAIS prototype project are listed below:

- To provide a mechanism for investigating the DAIS architecture and system control procedures.
- To verify the DAIS architecture and the design of the DAIS hardware and software elements.
- To identify DAIS problem areas and to feedback problem information into the ongoing DAIS effort.
- To gain technical and management experience which is directly transferable to the DAIS development effort.

8.2 DAIS Prototype Overview

The DAIS Prototype was planned to be a hardware/software simulation of DAIS. This required that the hardware architecture and the system control procedures be developed according to DAIS baseline specification. However, since the specific performance characteristics of the processors in the system were not of critical importance, the DEC PDP 11/40 was selected as the DAIS Prototype processor. These minicomputers were readily available in the laboratory and possessed the required I/O features.

Multiplex Equipment - Since the operation of the data bus and the system protocol for communications using the data bus are critical to the system control, the BCI units for DAIS Prototype were designed directly to the DAIS BCI specification, and were built in breadboard form. As problems with the baseline specification were found during the breadboard development, changes to the baseline specification were made.

As in DAIS, the breadboard BCIs were used to connect the PDP 11/40's to the multiplex data bus. An RT/aircraft equipment combination was simulated by connecting an additional BCI to a separate PDP 11/40. The resulting DAIS Prototype system is shown in Figure 41; comparison to Figure 42 will show that the system architecture is identical to that of DAIS.

Mission Software - As previously stated, one of the objectives of the DAIS Prototype was to investigate the DAIS executive software baseline design. Therefore, an existing DAIS top level design specification for the DAIS executive software was used as the prototype executive baseline. For DAIS, the executive software will be developed in JOVIAL J73/I. Since a JOVIAL compiler was not available for the PDP 11/40, the prototype executive was written in assembly language using a set of macro constructs and a top down structured programming approach. This was done to insure transferability of the executive to DAIS.

The application software developed for the DAIS Prototype was a set of application stubs developed specifically to request executive services, to process received bus data, and to provide data for output on the bus.

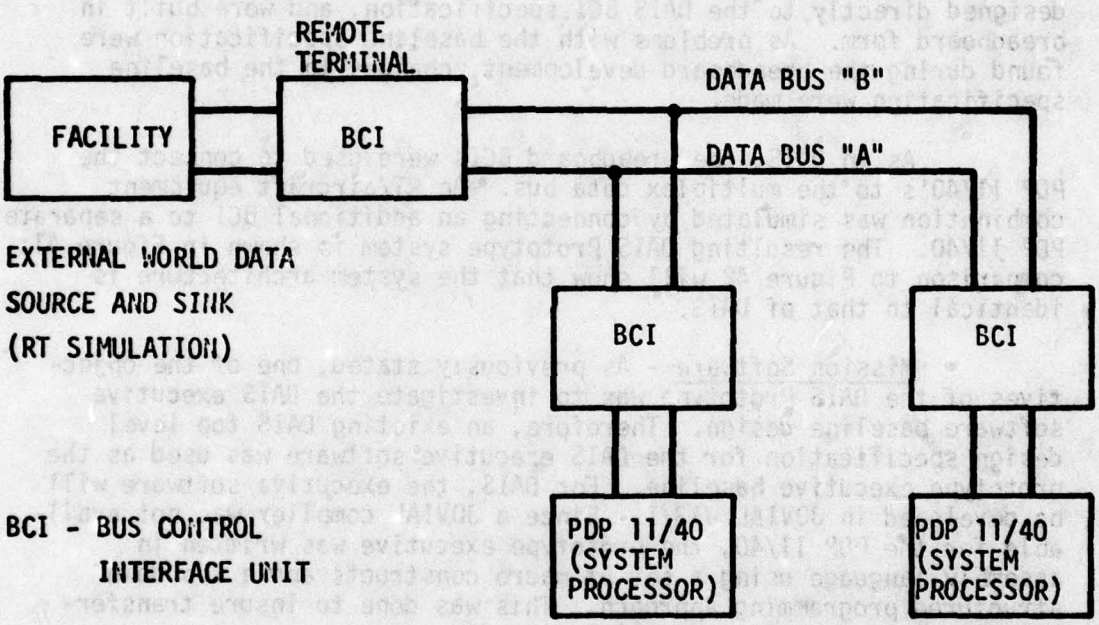


FIGURE 41. DAIS PROTOTYPE CONFIGURATION (FULL-UP)

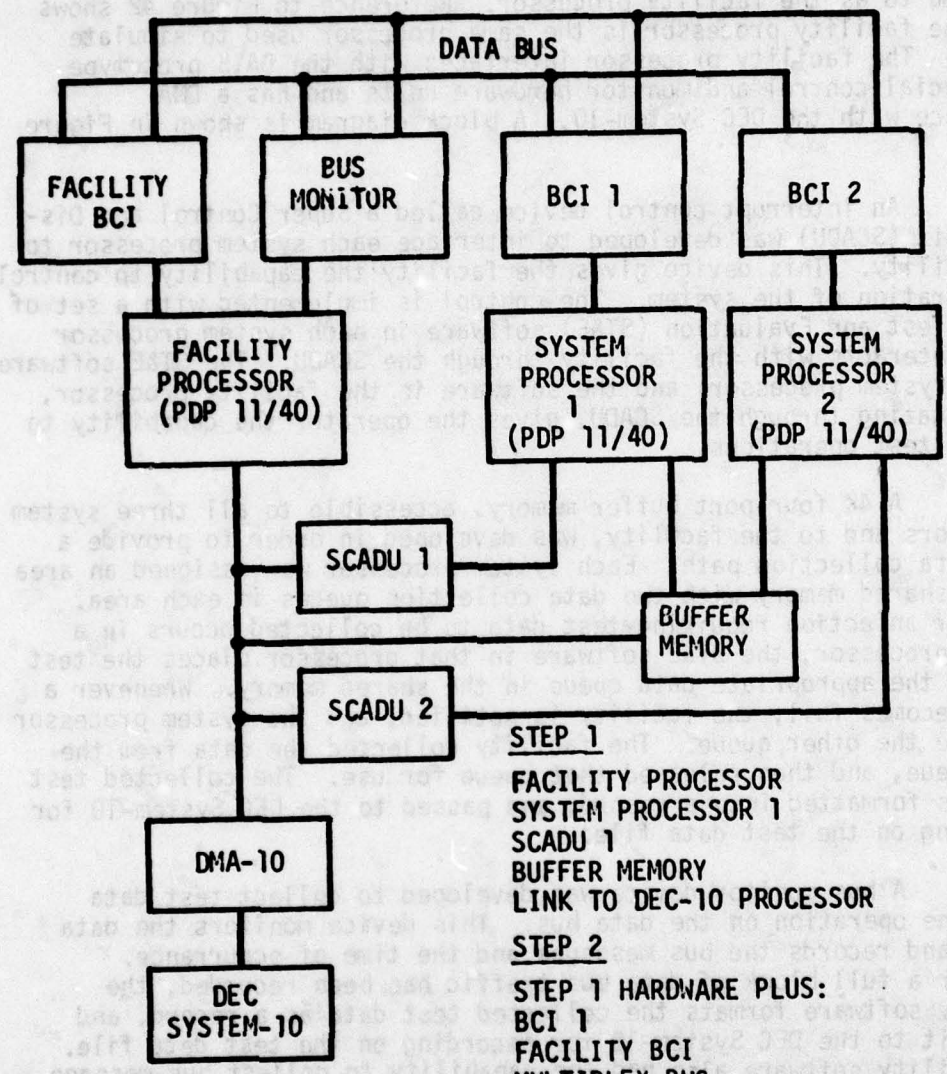


FIGURE 42. DAIS PROTOTYPE HARDWARE CONFIGURATION

DAIS Prototype Operation - To conduct simulation and testing activities on the DAIS prototype required a means of test control and monitoring. This capability was provided by a PDP 11/40 processor referred to as the facility processor. Reference to Figure 42 shows that the facility processor is the same processor used to simulate the RT. The facility processor interfaces with the DAIS prototype via special control and monitor hardware units and has a DMA interface with the DEC System-10. A block diagram is shown in Figure 42.

An interrupt control device called a Super Control and Display Unit (SCADU) was developed to interface each system processor to the facility. This device gives the facility the capability to control the operation of the system. The control is implemented with a set of System Test and Evaluation (ST&E) software in each system processor which interacts with the facility through the SCADU. The ST&E software in the system processors and the software in the facility processor, communicating through the SCADU, gives the operator the capability to control test operations.

A 4K four-port buffer memory, accessible to all three system processors and to the facility, was developed in order to provide a test data collection path. Each system processor was assigned an area of the shared memory with two data collection queues in each area. Whenever an action requiring test data to be collected occurs in a system processor, the ST&E software in that processor places the test data in the appropriate data queue in the shared memory. Whenever a queue becomes full, the facility is notified, and the system processor will use the other queue. The facility collected the data from the full queue, and then released that queue for use. The collected test data was formatted into a record, and passed to the DEC System-10 for recording on the test data file.

A bus monitor device was developed to collect test data about the operation on the data bus. This device monitors the data buses, and records the bus messages and the time of occurrance. Whenever a full block of data bus traffic has been recorded, the facility software formats the collected test data as a record, and passes it to the DEC System-10 for recording on the test data file. The facility software also had the capability to collect bus message data addressed to the facility's BCI.

The facility was interfaced to the DEC System-10 through a DMA channel. The facility used this link to pass test data records to the DEC System-10 for recording on the test data file. A test data collection routine ran on the DEC System-10 in real time, locked in core. This routine accepted each test data record and wrote it onto the test data file. This routine had a user interface which allows the operator to name the test data file at test initialization.

When a test data file was generated at the end of a test, the test data analysis program ran, using that test data file as an input. The operator examined the test data, printed specific portions of the data file, ran analysis routines on specific test data parameters, printed the results of the analysis, and printed the difference between the test data file and other test data files.

8.3 DAIS Prototype Integration and Test

There were three phases of activity associated with the DAIS Prototype project. During phase A, the different hardware peices were tested, design specifications were written for the software elements, and requirements were formulated for test control and monitoring. Phase B saw the integration of the hardware and software resulting in an operational prototype. During phase C a detailed evaluation of the DAIS prototype executive was performed with emphasis on measurement of the executive overhead.

Phase A - The hardware unit testing completed during phase A required the generation of special test software and, in most cases, required the integration of two or more pieces of the hardware. This not only assured the integrity of each hardware unit, but also exercised many of the electrical/functional connections between units.

While testing of the hardware units was being performed, the detailed requirements specifications for the executive was prepared using as a baseline existing top level specifications for the DAIS executive. The design of the executive was extended to incorporate features not covered in the existing specifications, especially in the area of system error and failure handling. A programming standards manual was written specifying development of the executive in a top down manner using the DEC MACRO assembler. The allowable program constructs were written as macros and added to the PDP 11/40 system macro library.

Also during phase A, requirements for the facility software and the ST&E software related to test control and monitoring were formulated. These requirements can be divided into four areas: test setup and control, data collection, error/failure simulation, and post test analysis.

The test setup and control functions allowed the test operator to preplan the test, specifying test length, bus data collection requirements, error/failure simulation actions, and minor cycle "freeze" points at which the test is to be stopped for investigative purposes. Interactive commands provided the operator with capabilities to continue from freeze points, to single step the system on a minor frame (8 msec) basis, and to stop the test.

The data collection function provided for collection of test data from several sources and on line transfer of the data to a file maintained on the DEC System-10. The sources from which data were collected are: 1) the bus monitor unit, which was programmed to collect all bus data for a short time period or selected bus data for longer time periods; 2) the facility's BCI, which process all bus messages addressed to the facility; and 3) the four-port buffer memory into which test data was stored by the ST&E software contained in the system processors. Refer to Figure 42 for the placement of these hardware units in the DAIS Prototype.

Error/failure simulation was performed by controlling the BCIs connected to the system processors, and the facility's BCI. Under operator control, either preplanned or interactive, any of these three units could be made to enter either a non-responding or busy state, or to simulate a total failure. Either of the first two states could be for a specified time or for the remaining test time.

The post test analysis capability provided by a DEC System-10 routine, allowed the test operator to dump all or specific sections of the data collected, to display portions of the data on a CRT, and to perform other analysis functions on the test data.

Phase B - During this phase, the hardware, the facility software, the ST&E software, the DAIS Prototype executive, and the application software stubs were integrated into a working system. Because of the number of pieces of hardware and software involved, a five step process was used to perform integration. Figure 42 depicts the hardware configuration for each step.

The objective of step 1 of phase B was to exercise the test control and monitor interfaces between the facility processor and one system processor, and to exercise some of the local executive functions. This required that the facility software and the ST&E software be operational. Portions of the executive which were tested were the local executive services of data read and write, and tasking control. Note that communication via the multiplex data bus was not tested during this step. Application tasks were simulated by stubs which were programmed to request executive services. Testing was performed, and test data was collected, in real time, from the executive. The test data was transferred via the multiport buffer memory to the facility processor and then to the DEC System-10. The data collected was examined to verify correct operation of the hardware and software.

In step 2, the multiplex data bus, the facility BCI, and the system processor BCI were added to the system. This provided a communication path between the system processor and an RT, as simulated by the facility processor. Those portions of the local executive which control synchronous bus communications were added to the step 1 executive. The capability to transmit and receive data in response to commands from the system processor was added to the facility software. Application stubs were used to request executive services, and test data was collected and written into multiport buffer memory by the ST&E software. Post test analysis was performed on the collected data to verify correct synchronous bus communications.

The second system processor was added in step 3, holding all the other hardware and software system variables constant. This configured the system to test synchronous bus communications between two system processors and a simulated RT. System processor 1 was designated the master processor, and thus contained the master executive

as well as a local executive. The synchronous bus control tables in the master processor were modified to control the transmission of messages between three terminals instead of the two terminals used in step 2.

Step 3 testing included making overhead measurements of all the executive services and the executive tasks associated with synchronous bus communications. These measurements showed the need for changes in the executive to reduce overhead. Since the executive was table driven, many of the changes were as simple as reordering of tables; for example, using unpacked tables to reduce the table decode time. On the other hand, the method of setting up for reception/transmission of synchronous messages was changed considerably.

The step 4 hardware configuration was identical to that of step 3. The capability to transmit asynchronous messages between system processors and between the system processors and the RT was incorporated into the executive. Testing was performed, and test data collected and analyzed to prove the correct operation of asynchronous message transmission. At completion of this step, the executive contained all its capabilities except that of error/failure recovery.

In step 5, the Bus Monitor Unit (BMU) which provides for monitoring of message data on the bus was added to the system. In the software area, step 5 was a significant change since it saw the addition of the error/failure recovery capability. Error recovery involves repeating of the bus message over the same or a redundant path. Recovery from equipment failures requires the identification of the failure and the use of a redundant path for transmission of the message, or the isolation of the failed equipment when the failure involves both redundant elements. The DAIS error/failure handling approach was used as a baseline for the equivalent DAIS Prototype capability.

As part of step 5, the executive was also updated to incorporate the efficiency related changes identified during step 3. Testing was then performed to verify the error/failure recovery capability and to insure that the efficiency related changes were made properly. In testing the error/failure capability of the executive, extensive use was made of the facility's capability to simulate transient and permanent failures in the three BCIs.

Phase C - The objective of this phase was a detailed evaluation of the executive. As part of the DAIS project itself, a series of close air support (CAS) missions are planned. A detailed analysis of one of the CAS missions has been performed to determine the expected bus loading and the expected load on the master and local executive. For phase C, the number of simulated tasks in each system processor was as expected in the CAS mission, the simulated application task was programmed to request executive service at the rate expected, and the expected number of synchronous and asynchronous bus messages was

transmitted. Testing was performed to obtain overhead measurements of executive bus message control tasks, executive control of application tasks, and all executive services provided to the application software. The information obtained was used to identify executive problem areas and to approximate the overhead of the DAIS executive.

8.4 DAIS Prototype Conclusions

The DAIS Prototype project was organized to provide transfer of information and experience to the DAIS. DAIS core elements specifications were used in development of the equivalent DAIS Prototype core elements. Key personnel were assigned similar areas of responsibility on both projects; and "Design To" and "As Designed" specifications were published for the hardware and software elements. Much of the prototype project experience was directly transferable to the DAIS effort resulting in problem corrections and improved designs for the DAIS system elements. Some of the more important benefits were:

1. Hardware Changes - In the process of interfacing the executive software to the BCIs, problem areas of the BCI were uncovered, and other areas where improvements could be made were identified. Consideration of the BCI brought about a reconsideration of the RT design where a message transfer problem was discovered and solved.

2. Executive Design - The DAIS Prototype executive, plus recommended changes resulting from the evaluation of the executive, became the baseline for the DAIS executive.

3. System Control Procedures - Development of the DAIS Prototype executive, especially in the areas of asynchronous communications, contributed to development of the DAIS system control procedures. Likewise, incorporation of the error/failure handling capability into the prototype executive provided an evaluation of the DAIS system control procedures in a time frame which allowed for modifications.

Benefits also resulted from prototype activities not directly related to evaluation of core elements and system operation. These are summarized below:

1. In developing the executive, a top down, structured programming methodology applicable to real time assembly language programming was developed and successfully used. This methodology and experience will prove useful in development of the support software for the DAIS.

2. The techniques developed for test control and monitoring will contribute to the development of similar capabilities for the DAIS test system.

9.0 CONCLUSION AND RECOMMENDATIONS

The DAIS program successfully developed and tested the DAIS architecture using representative core elements, thus demonstrating a solution to the problem of proliferation and nonstandardization of aircraft avionics. The key results of the program included:

- An architecture which is adaptable to various avionic applications.
- Demonstrated multiple configuration of the architecture without modifying the architecture - e.g. open ended system capability
- Technology for the standardization of common, interchangeable, and shared system core elements and support elements.
- Set of verified specifications for the DAIS architecture, core elements, and support elements.*

The results of the DAIS program will permit the Air Force to assume the initiative in the specification of avionic systems for acquisition of new or retrofit weapon systems. This will result in significant payoffs and benefits for the acquisition, operation and logistics of weapon systems as summarized below:

a. Acquisition

- Reduced proliferation of system and subsystems, and associated support elements.
- Validated systems and hardware/software specification
- Procurement of interchangeable subsystem elements
- Economies in production

b. Operational Benefits

- Increased ability to maintain avionics systems in an operational ready state.
- Ability to respond to new or changing threats.
- Increased maturity of common elements.
- Increased system availability.

*The DAIS Specifications are identified in MA 100 100, DAIS Document Description Manual.

c. Logistics Benefits

- Family of interchangeable subsystems and associated support elements
- Common support tools (software and hardware)
- Reduced retrofit costs by dictating standard interfaces, dictating the use of common and shared system elements and programming standards and language
- Reduced training and maintenance costs for common elements
- On-board tests to reduce maintenance costs.

The net results will be reduced life cycle costs of avionics (e.g. reduced total system costs), and improved availability (e.g. increased sortie generation rate and mission readiness) as well as ability to adapt to new threats.

The DAIS program has demonstrated that architecture, interface and core element standards can form the building blocks for the configuration of avionic systems which will avoid proliferation of development programs and their costs, and result in significant reductions in support and maintenance costs. Significant benefits can be achieved by the Air Force in the continued promotion and development of these DAIS concepts. The task that now needs to be accomplished is the transition of DAIS and related technology in the form of standards which can be applied to new aircraft avionics developments and retrofit or upgrade programs.

*U.S.Government Printing Office: 1979 — 657-002/526